

Architetture dei Sistemi a Elaborazione – a.a. 2010/11

Esercitazione di Laboratorio 3

1. Si scriva un programma in linguaggio Assembler 8086 che esegua le seguenti operazioni:
 - a. Acquisisca da tastiera una stringa di caratteri
 - b. Controlli se la stringa acquisita è palindroma
 - c. Visualizzi a video un messaggio
 - i. che indichi se la stringa se è palindroma o meno
 - ii. nel caso non sia palindroma stampi a video la stringa invertita.
2. Si scriva un programma in linguaggio Assembler 8086 che esegua le seguenti operazioni:
 - a. Riceva da tastiera due numeri positivi con valore massimo 10000
 - b. Se si tratta di numeri validi, ne esegua la media
 - c. Visualizzi il risultato a video

n.b.: acquisire un numero da tastiera significa riceverlo in caratteri ASCII, dopodiché convertirlo tramite moltiplicazioni successive per 10. Ad esempio “123” è composto da 3 caratteri ASCII, ‘1’ ‘2’ e ‘3’, per ottenere 123D occorre prima convertire le singole cifre (i.e., ‘1’ \rightarrow 1D), poi eseguire il seguente calcolo $\rightarrow (((1*10)+2)*10)+3$.

3. Si scriva un programma in linguaggio Assembler 8086 che esegua un’operazione di filtraggio di una serie di segnali.
 - a. Si definiscano due vettori di elementi su 16 bit chiamati `campioni` e `serie` (si assuma che il numero di elementi sia compreso tra 1 e 100).
 - b. Eseguire un’operazione di filtraggio sul contenuto di `campioni`, corrispondente a calcolare per ciascun campione il valore medio di quello precedente, del successivo e del valore stesso.
 - c. memorizzare gli `n` campioni dopo il filtraggio nel vettore `serie`.

Si strutturi il programma in modo che l’operazione di filtraggio sia eseguita invocando una procedura chiamata `filtro`. Questa procedura restituisce al chiamante il valor medio di tutti gli elementi in `campioni` e deve sfruttare il meccanismo di passaggio parametri via STACK.

Per il primo e l’ultimo campione si esegua la media su 2 valori.

Esempio

Si supponga che il vettore `campioni` contenga

5
41
27
345
9
190

Il vettore `serie` conterrà

23
24
137
127

181
99

```
; Si scriva un programma in linguaggio Assembler 8086 che esegua le seguenti operazioni:
; a. Acquisisca da tastiera una stringa di caratteri
; b. Controlli se la stringa acquisita è palindroma
; c. Visualizzi a video un messaggio
; i. che indichi se la stringa se è palindroma o meno
; ii. nel caso non sia palindroma stampi a video la stringa invertita.
```

```
LEN EQU 13
```

```
.model small
.stack
.data
```

```
STRING      db LEN DUP(0FFh)
MESSAGGIO1  db "La stringa e' palindroma", 13, 10, '$' ; CR/LF
MESSAGGIO2  db "La stringa non e' palindroma", 13, 10, '$'
```

```
.code
.startup
```

```
lea dx, STRING
mov byte ptr STRING, LEN-2
mov ah, 0Ah
int 21h
sub sp, 2 ; per il valore di ritorno
call CHECK_PAL
pop ax ; in ax ho il valore di ritorno della funzione
cmp ax, 0
jz no_palindroma
lea dx, MESSAGGIO1
mov ah, 9
int 21h
jmp fine
```

```
no_palindroma: lea dx, MESSAGGIO2
mov ah, 9
int 21h
mov si, 2
mov cl, byte ptr STRING[1]
mov ch, 0
add si, cx
dec si ; si punta alla fine della stringa
mov ah, 2
```

```
stampa: mov dl, STRING[si]
int 21h
dec si
loop stampa
mov dl, 13
int 21h
mov dl, 10
int 21h
jmp fine
```

```
CHECK_PAL proc near

push bp
mov bp, sp
mov [bp+4], word ptr 0
push ax
push bx
push cx
xor ax, ax
xor bx, bx
xor cx, cx
mov si, 2 ; si indice al primo elemento
mov cl, byte ptr STRING[1] ; ho in cl il numero di caratteri immessi
mov ch, 0
add bx, 2
add bx, cx
dec bx ; bx indice dell'ultimo carattere
```

```

        shr cx, 1          ; in cx ho il numero di iterazioni

ciclo:   mov al, STRING[si]
        cmp al, byte ptr STRING[bx]
        jnz exit
        inc si
        dec bx
        loop ciclo
        mov [bp+4], word ptr 1

exit:    pop cx
        pop bx
        pop ax
        pop bp
        ret

CHECK_PAL    endp

fine:       nop

.exit
end

```

```
; Si scriva un programma che segua le seguenti operazioni:
; a. Riceva da tastiera due numeri positivi con valore massimo 10000
; b. Se si tratta di numeri validi, ne esegua la media
; c. Visualizzi il risultato a video
; n.b. acquisire un numero da tastiera significa riceverlo in caratteri ASCII,
; dopodichè convertirlo tramite moltiplicazioni successive per 10.
; Ad esempio "123" è composto da 3 caratteri ASCII, '1', '2' e '3', per ottenere
; 123D occorre prima convertire le singole cifre (i.e. '1' -> 1D), poi eseguire
; il seguente calcolo -> ((1*10)+2)*10)+3.
```

```
LEN EQU 6
```

```
.model small
.stack
.data
```

```
MEDIA          dw 0
MEDIA_STR      db LEN dup(0)
NUMERO         db LEN, 0, LEN dup(0)
STRINGA1       db 13, 10, "Inserire numero: $"
STRINGA2       db 13, 10, "Numero non valido!", 13, 10, '$'
STRINGA3       db 13, 10, "Media: $"
```

```
.code
.startup
```

```
mov cx, 2
```

```
ciclo:      mov ah, 9
            lea dx, STRINGA1
            int 21h
            mov ah, 0ah
            lea dx, NUMERO
            int 21h
            sub sp, 2          ; per il valore di ritorno
            call CHK_STRING
            pop ax ; se ax != 0, il numero presente e' valido e vale proprio ax
            cmp ax, 0
            jne next          ; il numero è valido
            mov ah, 9
            lea dx, STRINGA2
            int 21h
            jmp ciclo
```

```
next:      add MEDIA, ax
            loop ciclo
            mov ax, MEDIA
            shr ax, 1          ; divido per 2, faccio la media
            mov MEDIA, ax
            jmp stampa_media
```

```
CHK_STRING proc near

            push bp
            mov bp, sp
            mov [bp+4], word ptr 0 ; azzero il valore di ritorno
            push ax
            push bx
            push cx
            push dx
            push si

            xor ax, ax
            mov bx, 10
            mov ch, 0
            mov cl, NUMERO[1] ; cx contiene il numero di caratteri immessi
            mov si, 2          ; si punta al primo carattere

            cmp cx, 1
            jnz get_num
```

```

        mov al, NUMERO[2]
        sub ax, '0'
        mov [bp+4], ax      ; aggiorno il valore di ritorno
        jmp cont

get_num:    mov al, NUMERO[si]
            sub ax, '0'
            mul bx
            dec cx

ripeti:    inc si
            mov dl, NUMERO[si] ; mettere dx e byte ptr
            sub dx, '0'
            add ax, dx
            xor dx, dx
            dec cx
            jz fine_num
            mul bx
            jmp ripeti

fine_num:  cmp ax, 10000
            ja cont; ax > 10000 quindi [bp+4], il valore di ritorno, rimane 0
            mov [bp+4], ax      ; aggiorno il valore di ritorno

cont:      pop si
            pop dx
            pop cx
            pop bx
            pop ax
            pop bp
            ret

CHK_STRING    endp

        stampa media in esadecimale
stampa_media: mov ah, 9
            lea dx, STRINGA3
            int 21h

            mov ch, 4 ; contatore
            mov cl, 12 ; numero shift ad ogni ciclo
            mov bx, MEDIA

ciclo2:    mov ax, bx
            and ax, 0f000h
            shr ax, cl
            add ax, '0'
            cmp ax, '9'
            jbe print_it
            add ax, 'A'-'0'-10

print_it:  mov dl, al
            mov ah, 2
            int 21h
            mov cl, 4
            rol bx, cl
            mov cl, 12
            dec ch
            jnz ciclo2

            mov dl, 'h'
            int 21h
            mov dl, ' '
            int 21h
            mov dl, '-'
            int 21h
            mov dl, '>'
            int 21h
            mov dl, ' '
            int 21h

```

```

mov ax, MEDIA
xor cx, cx ; Quanti numeri visualizzo
mov dx, 0
mov bx, 10

ciclo3:    div bx      ; ax = [dx,ax]/bx, dx = [dx,ax]%bx
           add dl, '0'
           push dx
           inc cx
           xor dx, dx
           cmp ax, 0
           jnz ciclo3

           mov ah, 2
stampa:    pop dx
           int 21h
           loop stampa

           mov dl, 'D'
           int 21h
           mov dl, 13
           int 21h
           mov dl, 10
           int 21h

.exit
end

```

```

;Si scriva un programma che esegua un'operazione di filtraggio di una
;serie di segnali.
;   a. Si definiscano due vettori di elementi su 16 bit chiamati campioni
;       e serie (si assuma che il numero di elementi sia compreso tra 1 e 100)
;   b. Eseguire un'operazione di filtraggio sul contenuto di campioni,
;       corrispondente a calcolare per ciascun campione il valore medio di quello
;       precedente, del successivo e del valore stesso.
;   c. Memorizzare gli n campioni dopo il filtraggio nel vettore serie.
;Si strutturi il programma in modo che l'operazione di filtraggio sia eseguita
;invocando una procedura chiamata filtro. Questa procedura restituisce al
;chiamante il valor medio di tutti gli elementi in campioni e deve sfruttare
;il meccanismo di passaggio parametri via STACK.
;Per il primo e l'ultimo campione si esegua la media su 2 valori

```

```

        LEN equ 6

```

```

.model small
.stack
.data

```

```

        CAMPIONI dw 5, 41, 27, 345, 9, 190
        SERIE     dw LEN dup (0ffh)

```

```

.code
.startup

```

```

        lea dx, CAMPIONI
        mov cx, LEN
        mov si, 0

```

```

ciclo:   cmp si, 0
         jz confine_sx
         cmp si, 2*LEN-2
         jz confine_dx
         sub sp, 2 ; valore di ritorno
         push CAMPIONI[si-2]
         push CAMPIONI[si]
         push CAMPIONI[si+2]
         mov ax, 3
         push ax      ; di quanti numeri devo calcolare la media
         call filtro
         jmp next

```

```

confine_sx: sub sp, 2 ; valore di ritorno
            push CAMPIONI[si]
            push CAMPIONI[si+2]
            mov ax, 2
            push ax      ; di quanti numeri devo calcolare la media
            call filtro
            jmp next

```

```

confine_dx: sub sp, 2
            push CAMPIONI[si-2]
            push CAMPIONI[si]
            mov ax, 2
            push ax
            call filtro

```

```

next:     pop ax ; ho la media, presa dal valore di ritorno della funzione filtro
         mov SERIE[si], ax
         add si, 2
         loop ciclo
         jmp fine

```

```

filtro    proc near

         push bp
         mov bp, sp
         push ax
         push bx

```



```

    mov ax, [bp+4]      ; ax contiene quanti numero devo mediare (2 o 3)
    cmp ax, 2
    jz media_2

    mov ax, [bp+10]
    add ax, [bp+8]
    add ax, [bp+6]
    mov bl, 3
    div bl
    mov ah, 0           ; in ax ho la media dei tre valori
    mov [bp+12], ax     ; scrivo il risultato come valore di ritorno
    pop bx
    pop ax
    pop bp
    ret 8

media_2:
    mov ax, [bp+8]
    add ax, [bp+6]
    shr ax, 1           ; in ax ho la media dei due valori
    mov [bp+10], ax     ; scrivo il risultato come valore di ritorno
    pop bx
    pop ax
    pop bp
    ret 6

filtro    endp

fine:     nop

.exit
end

```