

Esercizi Assembly 7

M. Sonza Reorda – M. Grosso

Politecnico di Torino
Dipartimento di Automatica e Informatica

Esercizio 1

- Nel calcolo combinatorio si definisce *combinazione semplice (senza ripetizioni)* una presentazione di elementi di un insieme nella quale non ha importanza l'ordine dei componenti e non si può ripetere lo stesso elemento più volte. Dati n elementi distinti e un numero intero positivo $k \leq n$, il numero di combinazioni semplici possibili $C(n, k)$ è dato dalla seguente formula:

$$C(n, k) = \binom{n}{k} = \frac{n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot (n-k+1)}{k!}$$

- Si scriva una procedura COMBINA in grado di calcolare il numero di combinazioni semplici dati i parametri n e k ricevuti come variabili globali di tipo *byte*. Il risultato dovrà essere restituito attraverso la variabile globale di tipo *word* risultato.
- Sia lecito supporre che durante le operazioni intermedie non si presenti *overflow*.
- Esempi:

– $n = 6$; $k = 3$	$C(n, k) = 20$
– $n = 12$; $k = 2$	$C(n, k) = 66$

Codice

```

.MODEL small
.STACK
.DATA
n      DB 6
k      DB 3
risultato DW ?
.CODE
.STARTUP
CALL COMBINA
.EXIT

COMBINA PROC
MOV CL, k
XOR CH, CH
DEC CX
MOV AL, n
XOR AH, AH
MOV BL, n

ciclo1: MUL BX
DEC BX
LOOP ciclo1
MOV BL, k
MOV CL, k
XOR CH, CH
DEC CX

ciclo2: DIV BL
SUB BL, 1
LOOP ciclo2
MOV risultato, AX
RET
COMBINA ENDP
END

```

Esercizio 2

- Scrivere un programma che stampi su video una stringa per la richiesta di introduzione di un numero intero, legga il numero e stampi su video un messaggio che specifichi se il numero che è stato introdotto è pari o dispari
- Si verifichi la corretta digitazione da parte dell'utente e si segnalino eventuali errori (es., introduzione di caratteri alfabetici)
- Si usi per la stampa la procedura come definita nell'Esercizio 2 dell'Esercitazione 6.

Codice

```

CR      EQU 13
NL      EQU 10
LUNG_MSG EQU 30
        .model small
        .stack
        .data
aa      db 2
bb      db 4
cc      db 2
intro_msg db "Introdurre intero e ENTER: ", CR, NL
pari_msg  db " Numero inserito ----> pari", CR, NL
disp_msg  db " Numero inserito --> dispari", CR, NL
err_msg   db " Digitazione non corretta!!!", CR, NL
        .code
        .startup
        lea ax, intro_msg
        mov bx, LUNG_MSG
        call stampa

```

Codice [cont.]

```

        mov bx, 0    ; conto cifre acquisite
        mov ah, 1
leggi:   mov dl, al
        INT 21h
        cmp al, CR
        je next
        cmp al, '0'
        jl errore
        cmp al, '9'
        jg errore
        inc bx
        jmp leggi
next:    cmp bx, 0
        je errore
        sub dl, '0'
        mov bx, LUNG_MSG

```

Codice [cont.]

```

    test dl, 1
    jz pari
    lea ax, disp_msg
    call stampa
    jmp fine
pari:    lea ax, pari_msg
    call stampa
    jmp fine

errore:  mov bx, LUNG_MSG
    lea ax, err_msg
    call stampa
    jmp fine

fine:    .exit

```

(uso procedura **stampa** come definita nell'Esercizio 2 dell'Esercitazione 6)

Esercizio 3

- Un *indirizzo IP* è un numero che identifica univocamente un dispositivo collegato a una rete che utilizza *Internet Protocol* come protocollo di comunicazione. L'*Internet Protocol version 4* (IPv4) prevede che l'indirizzo sia costituito da 32 bit (4 byte) suddivisi in 4 gruppi da 8 bit (1 byte), separati ciascuno da un punto. Ciascuno di questi 4 byte è poi convertito in formato decimale di più facile interpretazione. Un esempio di indirizzo IPv4 è 130.192.182.133, che corrisponde a 82C0B685h.
- Si scriva una procedura **filtro** in grado di elaborare una sequenza di indirizzi IPv4 e contare quanti di essi soddisfino la seguente condizione: l'indirizzo deve essere confrontato bit a bit con un *riferimento* dato, ma nel confronto devono essere considerati soltanto i bit nelle posizioni che, in una variabile *doubleword maschera*, hanno valore corrispondente a '1'. Se i bit confrontati corrispondono, la condizione è soddisfatta. Viceversa, i bit nelle posizioni corrispondenti a valori '0' nella maschera non devono essere considerati per il confronto.
- Esempio:

Riferimento	82C0B685h	10000010.11000000.10110110.10000101
Maschera	FFFC0000h	11111111.11111100.00000000.00000000
Indirizzo 1	82C028D1h	10000010.11000000.00101000.11010001
→	soddisfa requisiti	
Indirizzo 2	81C0276Ah	10000001.11000000.00100111.01101010
→	non soddisfa requisiti.	

Esercizio 3^[cont.]

- Sia dato in memoria un vettore *address* di *doubleword* contenente la sequenza di indirizzi IPv4 (la dimensione del vettore è pari a DIM, assegnata come costante). Sia data inoltre una variabile *doubleword* *mask* contenente la maschera. La procedura riceve come parametri, tramite lo *stack*, l'indirizzo di riferimento per il confronto e gli indirizzi di *address* e *mask*, e restituisce il numero di elementi che soddisfano la condizione sempre attraverso lo *stack*.

- Esempio di programma chiamante:

```

push 82C0h      ; parte alta di indirizzo di riferimento
push 0B685h    ; parte bassa di indirizzo di riferimento
lea AX, address
push AX
lea AX, mask
push AX
sub SP, 2      ; spazio riservato per risultato
call filtro
pop AX         ; prelevamento risultato da stack
add SP, 8

```

Codice

```

DIM      EQU 8
.MODEL small
.STACK
.DATA
address DD 82C0051AH, 0C0A80A01H, 4A7D276AH, 0D5FE1150H
        DD 0C7EF88C8H, 82C0B621H, 82C0A4F5H, 0ADC01874H
mask    DD 0FFFF0000H
.CODE
.STARTUP
push 82C0h      ; parte alta di indirizzo di riferimento
push 0B685h    ; parte bassa di indirizzo di riferimento
lea AX, address
push AX
lea AX, mask
push AX
sub SP, 2      ; spazio riservato per risultato
call filtro
pop AX         ; prelevamento risultato da stack
add SP, 8
.EXIT

```

Codice [cont.]

```

filtro  PROC
        MOV BP, SP
        MOV SI, [BP+6]    ; indirizzo address
        MOV DI, [BP+4]    ; indirizzo mask
        XOR BX, BX
        MOV CX, DIM
ciclo:  MOV AX, WORD PTR [SI]
        XOR AX, [BP+8]
        AND AX, WORD PTR [DI]
        JNZ next
        MOV AX, WORD PTR [SI+2]
        XOR AX, [BP+10]
        AND AX, WORD PTR [DI+2]
        JNZ next
        INC BX
next:   ADD SI, 4
        LOOP ciclo
        MOV [BP+2], BX
        RET
filtro  ENDP
        END

```

Esercizio 4

- Si scriva un programma che, data una stringa di lunghezza nota in memoria, calcoli la frequenza di ciascuna lettera e stampi quindi a video un istogramma orizzontale delle frequenze rilevate (omettendo i caratteri a frequenza nulla).
 - Esempio: “sasso rosso”

```

a *
o ***
r *
s *****

```

Implementazione

- Prima fase: calcolo delle frequenze di ciascun carattere (ipotesi: solo caratteri minuscoli)
 - Scarto caratteri non alfabetici
 - Uso un vettore di frequenze (26 caratteri)
- Seconda fase: stampa dell'istogramma.

Codice

```
DIM      EQU 11
CR       EQU 13
NL       EQU 10
        .model small
        .stack
        .data
stringa  db "sasso rosso"
freq     db 26 DUP (0)
        .code
        .startup
        mov cx, DIM
        mov ax, 0
        mov si, 0
ciclo:   mov al, stringa[si]
        cmp al, 'a'
        jl next
        cmp al, 'z'
        jg next
```

Codice [cont.]

```

        sub al, 'a'
        mov di, ax
        inc freq[di]
next:    inc si
        loop ciclo

        mov cx, 26
        mov si, 0
ciclo2:  call stampa_star
        inc si
        loop ciclo2
        .exit

```

Codice [cont.]

```

stampa_star proc
    push ax
    push cx
    push dx
    xor cx, cx
    cmp freq[si], 0
    jz fine_star
    mov cl, freq[si]
    mov ah, 2
    mov dl, CR
    INT 21h
    mov dl, NL
    INT 21h
    mov dx, 'a'
    add dx, si
    INT 21h
    mov dl, ' '
    INT 21h
    mov dl, '*'

```


Codice [cont.]

```
ciclo_star: INT 21h
            loop ciclo_star
fine_star:  pop dx
            pop cx
            pop ax
            ret
stampa_star endp

            end
```