Esercizio di programmazione

sino a 12 punti – è possibile consultare qualunque materiale cartaceo - tempo: 60 minuti

Siano date due matrici di interi positivi memorizzate per righe:

Matrice 1:

$riga\ 0$:	1	2	3	4
$riga\ 1$:	5	6	**7**	8
$riga\ 2$:	9	10	11	12
$riga\ 3$:	13	14	15	16

Matrice 2:

$riga\ 0$:	9	3	4	11
$riga\ 1$:	1	5	6	8
$riga\ 2$:	16	15	2	12
$riga\ 3$:	**7**	10	13	14

Data l'indicazione dell'indice di una riga per ciascuna matrice (ad esempio, 1 e 3, rispettivamente), si scriva in linguaggio Assembly 8086 una procedura `trova_num` in grado di determinare il valore dell'elemento presente in ciascuna delle due matrici alle righe indicate (in questo caso, 7).

La procedura riceve mediante lo *stack* gli indirizzi di due vettori di `DIM`x`DIM` *byte* (`DIM` dichiarato come costante) contenenti le due matrici, e gli indici di riga mediante AL e AH. Il risultato, ossia il valore trovato, deve essere restituito tramite lo *stack*.

Se non ci sono elementi in comune, la procedura deve restituire il valore -1. Se ci sono più elementi in comune, la procedura ne restituisca uno qualunque fra essi.

Di seguito un esempio di programma chiamante:

```
DIM         EQU 4
            .model small
            .stack
            .data
mat1        db  1,  2,  3,  4
            db  5,  6,  7,  8
            db  9, 10, 11, 12
            db 13, 14, 15, 16
mat2        db  9,  3,  4, 11
            db  1,  5,  6,  8
            db 16, 15,  2, 12
            db  7, 10, 13, 14

            .code
            .startup
            push offset mat1
            push offset mat2
            sub sp, 2
            mov al, 1       ; indice riga mat1
            mov ah, 3       ; indice riga mat2
            call trova_num
            pop ax          ; risultato
            add sp, 4
            .exit
```

Soluzione proposta

```
DIM         EQU 4

            .model small
            .stack
            .data
mat1        db  1,  2,  3,  4
            db  5,  6,  7,  8
            db  9, 10, 11, 12
            db 13, 14, 15, 16
mat2        db  9,  3,  4, 11
            db  1,  5,  6,  8
            db 16, 15,  2, 12
            db  7, 10, 13, 14

            .code
            .startup
            push offset mat1
            push offset mat2
            sub sp, 2
            mov al, 1       ; indice riga mat1
            mov ah, 3       ; indice riga mat2
            call trova_num
            pop ax          ; risultato
            add sp, 4
            .exit

trova_num   proc
            push bp         ; salvataggio del contenuto dei registri
            mov bp, sp
            push ax
            push bx
            push cx
            push si
            push di

            mov di, [bp+6]  ; prelevamento dati dallo stack
            mov si, [bp+8]

            mov cx, ax
            mov ah, DIM
            mul ah
            add si, ax
            mov al, ch
            mov ah, DIM
            mul ah
            add di, ax

            mov cx, DIM
            mov bx, di
ciclo1:     push cx
            mov cx, DIM
            mov di, bx
            mov al, [si]
ciclo2:     cmp al, [di]
            je fine
            inc di
            loop ciclo2
```

```
            inc si
            pop cx
            loop ciclo1

            push cx         ; per garantire equilibrio tra push e pop
            mov al, -1

            fine:
            pop cx
            mov ah, 0       ; salvataggio del risultato nello stack
            mov [bp+4], ax

            pop di          ; ripristino del contenuto dei registri
            pop si
            pop cx
            pop bx
            pop ax
            pop bp
            ret

trova_num   endp

            end
```