

## Esercizio di programmazione

sino a 12 punti – è possibile consultare solamente l'Instruction Set Intel - tempo: 60 minuti

Sia data una matrice di byte contenente solo valori 0 e 1, di dimensione fissata dalle due costanti (strettamente positive) N<sub>RIGHE</sub> e N<sub>COLONNE</sub>. Le celle contenenti il valore 1 corrispondono ai punti di una linea (o percorso) sul piano. Si scriva una procedura **seguiPercorso** in linguaggio Assembly 8086 in grado di seguire un percorso a partire da una cella data e lungo celle contigue contenenti il valore 1, finché possibile.

Il percorso non presenta biforcazioni, e lo spostamento può avvenire soltanto verso destra o verso il basso. Nell'esempio seguente, la casella di partenza è  $(1, 0)$ , il percorso è lungo 6 celle e la casella finale è  $(3, 3)$ :

	0	0	0	0	1	1
inizio	1	1	0	0	0	1
	0	1	1	1	0	0
	0	0	0	1	0	1
				fine		

La procedura riceve:

- l'offset della matrice tramite stack
- l'indice della riga di partenza (compreso tra 0 e NGRIGHE-1) attraverso il registro DL
- l'indice della colonna di partenza (compreso tra 0 e NCOLONNE-1) attraverso il registro DH.

La procedura restituisce la lunghezza del percorso tramite stack. Si noti che il percorso è lungo 0 se la cella di partenza ha valore 0.

Non è ammesso l'uso di variabili.

Di seguito un esempio di programma chiamante:

```
NRIGHE EQU 4
NCOLONNE EQU 6
.MODEL small
.STACK
.DATA
matrice DB 0, 0, 0, 0, 0, 0
         DB 1, 1, 0, 0, 0, 0
         DB 0, 1, 1, 1, 0, 0
         DB 0, 0, 0, 1, 0, 0
```

```
.CODE
.STARTUP
PUSH OFFSET matrice
SUB SP, 2
MOV DL, 1
MOV DH, 0
CALL seguiPercorso
POP AX
ADD SP, 2
.EXIT
```

## Soluzione proposta

```
seguiPercorso PROC
    PUSH BP                                ; salvataggio dei registri nello stack
    MOV BP, SP
    PUSH AX
    PUSH BX
    PUSH DX
    PUSH SI

    MOV AL, DH
    XOR AH, AH
    MOV SI, AX                            ; indice di colonna in SI
    MOV AL, NCOLONNE
    MUL DL
    MOV BX, [BP+6]                        ; offset della matrice
    MOV DX, BX
    ADD BX, AX                            ; offset + indice riga elemento iniziale
    ADD DX, NCOLONNE*(NRIGHE-1)          ; offset + indice ultima riga
    MOV WORD PTR [BP+4], 0
    CMP BYTE PTR [BX][SI],1
    JNE fine
    INC [BP+4]

ciclo:    CMP BX, DX                      ; controllo ultima riga
    JE next
    CMP BYTE PTR [BX+NCOLONNE][SI], 1    ; controllo elemento in basso
    JNE next
    INC WORD PTR [BP+4]
    ADD BX, NCOLONNE
    JMP ciclo

next:     CMP SI, NCOLONNE-1              ; controllo ultima colonna
    JE fine
    CMP BYTE PTR [BX][SI+1],1            ; controllo elemento a destra
    JNE fine
    INC WORD PTR [BP+4]
    ADD SI, 1
    JMP ciclo

fine:     POP SI
    POP DX
    POP BX
    POP AX
    POP BP
    RET
seguiPercorso ENDP

END
```

## Soluzione alternativa

```
seguiPercorso PROC
    PUSH BP
    MOV BP, SP
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX

    MOV BX, [BP+6]
    MOV AL, NCOLONNE
    MUL DL
    ADD BX, AX
    MOV AL, DH
    XOR AH, AH
    ADD BX, AX

    XOR CH, CH
    MOV CL, [BX]
    JCXZ fine

ciclo:    CMP DL, NRIGHE-1
          JE next
          CMP BYTE PTR [BX+NCOLONNE], 1
          JNE next
          ADD BX, NCOLONNE
          INC DL
          INC CX
          JMP ciclo

next:     CMP DH, NCOLONNE-1
          JE fine
          CMP BYTE PTR [BX+1], 1
          JNE fine
          INC BX
          INC DH
          INC CX
          JMP ciclo

fine:     MOV [BP+4], CX
          POP DX
          POP CX
          POP BX
          POP AX
          POP BP
          RET
seguiPercorso ENDP
```