

**Politecnico di Torino - III Facoltà di Ingegneria**  
**Esercitazioni del corso "Sistemi a Microprocessore"**  
**Ing. Paolo Bernardi – email: [paolo.bernardi@polito.it](mailto:paolo.bernardi@polito.it)**

*Esercitazione 2 – Le istruzioni aritmetiche e per la manipolazione dei bit*

*Argomenti trattati:*

1. istruzioni aritmetiche.
2. istruzioni per la manipolazione dei bit.

**Esercizio 1:** Si scriva un programma in linguaggio Assembler 8086 che esegua la somma degli elementi di un vettore definito di tipo *doubleword*.

*Possibile svolgimento:*

```
.MODEL small
.STACK
.DATA
VETT DD 14, 356, 203, 66020, 1, 0
RES DD ?
.CODE
.STARTUP
LEA SI, VETT
MOV AX, 0
MOV BX, 0
MOV CX, 6
lab: ADD AX, [SI]
ADC BX, [SI+2]
ADD SI, 4
loop lab
MOV word ptr RES, AX
MOV word ptr RES+2, BX
.EXIT
END
```

**Esercizio 2:** Si scriva un programma in linguaggio Assembler 8086 che dica se un'equazione di secondo grado nella forma  $ax^2+bx+d=0$  ha o meno soluzione.

Formula risolutiva: 
$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

*Possibile svolgimento:*

```
.MODEL small
.STACK
.DATA
A DB 70
B DB 10
D DB 4
RES DD ?
SOL_MSG DB "Ha due soluzioni", 0DH, 0AH, "$"
NO_SOL_MSG DB "Non ha soluzioni", 0DH, 0AH, "$"
.CODE
.STARTUP
MOV AL, B
MUL B
CWD
MOV word ptr RES, AX
MOV word ptr RES+2, DX
```

```

MOV    AL,    A
MUL    D
MOV    CX,    4
MUL    CX
SUB    word ptr RES,    AX
SBB    word ptr RES+2,    DX
JB     no_sol
LEA    DX,    SOL_MSG
JMP    fine
no_sol: LEA    DX,    NO_SOL_MSG
fine:  MOV    AH,    09h
INT    21H
.EXIT
END

```

**Esercizio 3:** Si scriva un programma in linguaggio Assembler 8086 che trovi il valore massimo del comun divisore tra due numeri.

*Possibile svolgimento:*

```

.MODEL small
.STACK
.DATA
A      DB     5
B      DB     12
MAX    DB     ?
MIN    DB     ?
MCD    DB     ?
.CODE
.STARTUP
MOV    AH,    A
CMP    AH,    B
JG     amax
MOV    MIN,   AH
MOV    AH,    B
MOV    MAX,   AH
JMP    inizio
amax:  MOV    MAX,   AH
      MOV    AH,    B
      MOV    MIN,   AH
      XOR    CX,    CX
inizio: MOV    CL,    MIN
      MOV    AL,    MAX
      CBW
      DIV   CL
      CMP    AH,    0
      JE     trovato
      MOV    MAX,   CL
      MOV    MIN,   AH
      JMP    inizio
trovato: MOV    MCD,   CL
.EXIT
END

```

**Esercizio 4:** Si scriva un programma in linguaggio Assembler 8086 che, data una matrice quadrata di tipo *byte*, ne trasformi il contenuto moltiplicando e dividendo il valore di ogni cella per il proprio vicino destro e sinistro, rispettivamente. Se l'elemento è su un lato, sostituire il valore mancante con 1. Se un valore può provocare una operazione di divisione per 0, sostituirlo col valore 1. Si supponga che non si ecceda mai la precisione massima raggiungibile con il tipo *byte*.

<u>Esempio:</u>	1 2 4 5	→	2 8 10 1
	2 3 5 7	→	6 7 11 1
	3 4 6 8	→	12 8 12 1
	0 3 5 1	→	0 15 1 0

*Possibile svolgimento:*

```

lato      EQU    4
          .MODEL small
          .STACK
          .DATA
A         DB     1, 2, 4, 5
          DB     2, 3, 5, 7
          DB     3, 4, 6, 8
          DB     0, 3, 5, 1
B         DB     4 dup (?)
          DB     4 dup (?)
          DB     4 dup (?)
          DB     4 dup (?)
          .CODE
          .STARTUP
          MOV    SI, 0
          MOV    BX, 0
avanti:   MOV    AL, A[BX][SI]
          CMP    SI, 0
          JE     no_div
          CMP    SI, lato - 1
          JE     no_mul
          MUL    byte PTR A[BX][SI+1]
          CMP    A[BX][SI-1], 0
          JE     fatto
          DIV    byte PTR A[BX][SI-1]
fatto:    MOV    B[BX][SI], AL
          INC    SI
          JMP    avanti
no_div:   MUL    byte PTR A[BX][SI+1]
          JMP    fatto
no_mul:   CBW
          DIV    byte PTR A[BX][SI-1]
          MOV    B[BX][SI], AL
          ADD    BX, lato
          MOV    SI, 0
          CMP    BX, lato*lato
          JNE   avanti
          LEA   DI, A
          .EXIT
          END

```

**Esercizio 5:** Si scriva un programma in linguaggio Assembler 8086 che esegua una operazione di AND logico tra gli elementi di due vettori e ne memorizzi il risultato in un terzo vettore. Il programma deve inoltre contare gli elementi del vettore risultato del vettore a parità pari, cioè tali per cui il numero di 1 nella rappresentazione binaria del numero è pari.

*Possibile svolgimento:*

```

lung      EQU    4
          .MODEL small
          .STACK
          .DATA
A         DB     1, 2, 4, 5
B         DB     3, 4, 1, 3

```

```

RES      DB      lung dup (?)
PAR      DB      0
        .CODE
        .STARTUP
LEA     BX,     RES
LEA     SI,     A
LEA     DI,     B
MOV     CX,     lung
inizio:  MOV     AH, [SI]
        MOV     AL, [DI]
        AND     AL, AH
        JNP    no_par
        INC     PAR
no_par:  MOV     [BX], AL
        INC     BX
        INC     SI
        INC     DI
        LOOP   inizio
        .EXIT
        END

```

*Variante: si contino gli elementi del vettore risultato del vettore a parità pari*

```

lung     EQU 4
        .MODEL      small
        .STACK
        .DATA
A        DB      1, 2, 4, 5
B        DB      3, 4, 1, 3
RES      DB      lung dup (?)
PAR      DB      0
        .CODE
        .STARTUP
LEA     BX,     RES
LEA     SI,     A
LEA     DI,     B
MOV     CX,     lung
inizio:  MOV     AH, [SI]
        MOV     AL, [DI]
        AND     AL, AH
        TEST    AL, 00000001b
        JNZ    no_par
        INC     PAR
no_par:  MOV     [BX], AL
        INC     BX
        INC     SI
        INC     DI
        LOOP   inizio
        .EXIT
        END

```

**Esercizio 6:** Si scriva un programma in linguaggio Assembler 8086 che conti il numero di bit a 1 nella rappresentazione binaria di una variabile di tipo *byte*.

*Possibile svolgimento:*

```

lung     EQU 8
        .MODEL      small
        .STACK
        .DATA

```

```

A          DB      1
RES        DB      ?
           .CODE
           .STARTUP
           MOV     CX,    8
inizio:    ROL     A,    1
           JNC     zero
           ADD     RES,  1
zero:      LOOP   inizio
           .EXIT
           END

```

**Esercizio 7:** Si scriva un programma in linguaggio Assembler 8086 che calcoli la media tra due numeri di tipo *word* senza utilizzare l'istruzione DIV.

*Possibile svolgimento:*

```

lung      EQU     4
           .MODEL  small
           .STACK
           .DATA
A          DW     9
B          DW     19
MEDIA     DW     0
           .CODE
           .STARTUP
           MOV     AX,    A
           ADD     AX,    B
           SHR     AX,    1
           MOV     MEDIA,    AX
           .EXIT
           END

```