

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

Esercitazione 1 - Introduzione alla programmazione in Assembler 8086

Argomenti trattati:

1. Struttura di un programma Assembler.
2. Le direttive DB, DW e DD.
3. L'istruzione MOV.
4. Le istruzioni ADD e SUB.

Esercizio 1: Si scriva un programma in linguaggio Assembler 8086 che

- dichiarare una variabile VARB di tipo *byte*
- dichiarare una variabile VARW di tipo *word*
- dichiarare una variabile VARD di tipo *doubleword*
- inizializzare tutte le variabili a 0.

Possibile svolgimento:

```
                .MODEL small
                .STACK
                .DATA
VARB            DB    ?
VARW            DW    ?
VARD            DD    ?
                .CODE
                .STARTUP
MOV    VARB, 0
MOV    VARW, 0
MOV    word PTR VARD, 0
MOV    word PTR VARD+2, 0
                .EXIT
                END
```

Esercizio 2: Si scriva un programma in linguaggio Assembler 8086 che

- dichiarare un vettore di variabili di tipo *byte* chiamato VETT di 6 elementi
- inizializzare tutti gli elementi del vettore con la rispettiva posizione ordinale.

Esempio: VETT(0) = 0, VETT(1) = 1 VETT(6) = 6

Possibile svolgimento:

```
lung            EQU    6
                .MODEL small
                .STACK
                .DATA
VETT            DB    lung DUP(?)
                .CODE
                .STARTUP
MOV    SI, 0
MOV    CX, lung
MOV    BL, 0
ciclo:         MOV    VETT[SI], BL
                INC    BL
                INC    SI
                LOOP  ciclo
                .EXIT
                END
```

Esercizio 3: Si scriva un programma in linguaggio Assembler 8086 che date due variabili di tipo *word*, ne scambi il valore.

Possibile svolgimento:

```
                .MODEL small
                .STACK
                .DATA
VAR1            DW     5
VAR2            DW     6
                .CODE
                .STARTUP
                MOV    AX,    VAR1
                MOV    BX,    VAR2
                MOV    CX,    AX
                MOV    AX,    BX
                MOV    BX,    CX
                MOV    VAR2,  AX
                MOV    VAR1,  BX
                .EXIT
                END
```

! La seguente soluzione alternativa utilizza l'istruzione XCHG

```
                .MODEL small
                .STACK
                .DATA
VAR1            DW     5
VAR2            DW     6
                .CODE
                .STARTUP
                MOV    AX,    VAR1
                MOV    BX,    VAR2
                XCHG   AX,    BX
                MOV    VAR1,  AX
                MOV    VAR2,  BX
                .EXIT
                END
```

Esercizio 4: Si scriva un programma in linguaggio Assembler 8086 che scambi il contenuto di due vettori con elementi di tipo *byte*.

Possibile svolgimento:

```
                .MODEL small
                .STACK
                .DATA
VETT1           DB     0,1,2,3,4,5,6,7,8,9
VETT2           DB     9,8,7,6,5,4,3,2,1,0
                .CODE
                .STARTUP
                MOV    CX,    0AH
                MOV    SI,    0
lab:            MOV    AL,    VETT1[SI]
                XCHG   AL,    VETT2[SI]
                MOV    VETT1[SI],  AL
                INC    SI
                LOOP   lab
                .EXIT
                END
```

Esercizio 5: Si scriva un programma in linguaggio Assembler 8086 che esegua la somma di due numeri (su 8 bit) che si trovano nelle locazioni di memoria OP1 e OP2, e ponga il risultato nella locazione RIS.

Possibile svolgimento:

```

.MODEL small
.STACK
.DATA
OP1    DB    11
OP2    DB    7
RIS    DB    ?
.CODE
.STARTUP
MOV    AL,   OP1
ADD    AL,   OP2
MOV    RIS,  AL
.EXIT
END

```

Esercizio 6: Si scriva un programma Assembler 8086 che esegua la differenza fra due numeri (su 16 bit) che si trovano nelle locazioni di memoria OP1 e OP2, e ponga il risultato nella locazione RIS.

Possibile svolgimento:

```

.MODEL small
.STACK
.DATA
OP1    DW    30
OP2    DW    12
RIS    DW    ?
.CODE
.STARTUP
MOV    AX,   OP1
SUB    AX,   OP2
MOV    RIS,  AX
.EXIT
END

```

Esercizio 7: Si scriva un programma in linguaggio assembler che, dati due vettori VETT1 e VETT2 con elementi di tipo *byte*, ne esegua la somma degli elementi e memorizzi il risultato in un terzo vettore VETT3.

Esempio:

VETT1 = { 1, 2, 3, 4, 5, 6, 7 }
VETT2 = { 2, 2, 5, 1, 4, 7, 0 }
risultato → VETT3 = { 3, 2, 5, 4, 5, 7, 7 }

Possibile svolgimento:

```

lung    EQU    7
.MODEL small
.STACK
.DATA
VETT1   DB    1,2,3,4,5,6,7
VETT2   DB    2,2,5,1,4,7,0
VETT3   DB    lung DUP (?)
.CODE
.STARTUP
MOV    CX,   LENGHT VETT3
MOV    SI,   0

```

```
lab:    MOV    AL,   VETT1[SI]
        ADD    AL,   VETT2[SI]
        MOV    VETT3[SI], AL
        INC    SI
        LOOP   lab
        .EXIT
        END
```