Esercitazione – Istruzioni aritmetiche

Esercizio1 . (addizione)

Scrivere un programma in Assembly che esegua la somma tra due variabili di tipo byte e che memorizzi il risultato nel registro AL e poi nella variabile **ris.**
Modificare i valori delle due variabili come indicato sotto e vedere quali flags vengono modificati.

a) 127 e 1 [MODIFICATI: o=1 perché: , s=1, a=1]
127 + 1 è una SOMMA DI 2 POSITIVI, POSSO AVERE L'OVERFLOW

RANGE =[-2^{n-1},2^{n-1} -1] = [-128,127]=[80,7F]

	1		
127	7	F	
1	0	1	
0	8	0	**-128**
	1000	0000	

c	0
z	0
s	**1**
O	**1**
P	0
a	1

b) 128 e 1 [MODIFICATI: o=0 (perché?), s=1, p=1]

128+1=? **SOMMA TRA UN NUMERO NEGATIVO E POSITIVO, non posso avere overflow**

128	8	0	(Considerato **neg**)
1	0	1	
0	8	1	**-127**
	1000	0001	

c	**0**
z	0
s	**1**
O	**0**
P	1
a	0

Osservazione: 128 è considerato il numero negativo più piccolo; 129 il penultimo numero negativo ….. disposizione circolare dei numeri compresi nell'intervallo

[-2^{n-1},2^{n-1} -1] = [-128,127]=[80,7F]

c) 128 e −1 [MODIFICATI: o=1(perché?), c=1]

Esercizio 2. Sommare due numeri (es. num1=150d e num2= 200d) e **incrementare di 1 i**l risultato

```
.model small
.stack
.data
 num1 db 150
 num2 db 200
 ris  dw ?
.code
  inizio:
      mov ax,@data
      mov ds,ax
      mov bl,num1
      add bl,num2
      inc bx
      mov ris,bx
      mov ah,4ch
      int 21h
  end inizio
```

Esercizio 4. **S**ommare due numeri e **decrementare di 1** il risultato

Esercizio 5. Scrivere un programma in assembly che esegua la **differenza** fra due numeri (su 16 bit) che si trovano nelle locazioni di memoria (variabili) SOT1(=0A1h) e SOT2 (=42h) e ponga il risultato nella locazione RIS.

Esercizio 6. Scrivere un programma che calcoli l'espressione w=x-2y +3z (usare solo somme e differenze: 2y =y+y; 3z=z+z+z …). Si supponga che x,y,z e w siano numeri su 16 bit . Ad esempio usare i valori: x=100 (64h), y=45 (2Dh), z=15 (Fh). Risultato w=55 (37h)

Esercizio 7. Scrivere il seguente programma in assembly che esegue moltiplicazioni fra due numeri (su 8 bit) contenuti in due variabili. Con il debug (TD) cercare di capire la differenza tra MUL e IMUL.

```
; programma assembly che MOLTIPLICA due numeri della memoria centrale contenuti in due
;variabili

.model small
.stack
.data
        A Db 20
        M1 Db 5
        M2 DB -5
        F dw ?
.code
 inizio:
    ; prodotto tra due numeri byte senza segno
    mov ax,@data
    mov ds,ax

        mov al, A
        mov bl, M1
        mul bl
        mov F,ax

        ; prodotto tra due numeri con segno --> mul M2 è CONSIDERATO POSITIVO
        mov al, A
        mov bl, M2
        mul bl
        mov F,ax

        ; prodotto tra due numeri con segno --> imul M2 è CONSIDERATO NEGATIVO
        mov al, A
        mov bl, M2
        imul bl
       mov F,ax

        mov ah,4ch
        int 21h
 end inizio
```

Esercizio 8. Rifare l'es.6 usando l'operatore mul.

Esercizio 7. (moltiplicazione a 16 bit) Scrivere un programma in Assembly che esegua la moltiplicazione tra 65535 e 2. Con il debug verificare il risultato 1FFFE (DX=0001 e AX = FFFE)

```
.model small
.stack
.data

.code
 inizio:

   mov bx,65535
   mov ax,2          ; FFFF+FFFF=1FFFE
   mul bx            ; in debug verificare il risultato in DX:AX = 0001FFFE

   mov ah,4ch
   int 21h
 end inizio
```

Esercizio 7. Scrivere un programma in Assembly che esegua la media tra 10 e 20.

```
MOV AH, 10      ;Iniziamo inserendo nel registro AH il valore 10
MOV AL, 20      ;ed in AL il valore 20

ADD AL, AH      ;Sommiamo il contenuto di al con il contenuto di ah
MOV AH, 0       ;Ed azzeriamo AH, ora in AL sara' presente il valore 30

MOV BX, 2       ;Copiamo in BX il 2
DIV BL          ;E dividiamo il contenuto di AX per il contenuto di BL

MOV Ris, AL     ;Ed infine copiamo il risultato della divisione nella variabile dichiarata sotto.
                ;L'eventuale resto della divisione viene conservato nel registro AH in quanto si
                ;opera con dati di tipo Byte.
```

Esercizio 10. Eseguire in TD il seguente programma. Osservare che:

- ✓ mul **num2** viene memorizzato come
  mul **byte ptr [0005]**

OPERATORE **PTR**
sintassi: tipo PTR nome

forza l'assemblatore a modificare il tipo di dato avente come identificatore il nome della variabile. In caso di ambiguità (mul può essere un'operazione tra byte o word) bisogna specificare se il dato da manipolare è di tipo WORD (16 bit) o di tipo BYTE (8 bit), usando l'istruzione "PTR". Nel nostro caso **ptr** dice all'assemblatore che il valore della variabile num2 memorizzata nella locazione 0005 è un byte e quindi mul è un'operazione tra byte e non tra word.

- ✓ Quoto della divisione è posto in AL e il resto della divisione è posto in AH

```
; programma assembler che effettua le seguenti operazioni:
; moltiplicazione, divisione e resto

.model small
.stack
.data
 num1 db 50
 num2 db 16
 prodotto dw ?
 quoziente db ?
 resto db ?
.code
 inizio:
   mov ax,@data
   mov ds,ax
   mov al,num1
   mul num2
   mov prodotto,ax
   mov al,num1
   mov ah,0
   div num2
   mov quoziente,al
   mov resto,ah
   mov ah,4ch
   int 21h
 end inizio
```

Esercizio 9. (divisione a 16 bit) Scrivere un programma in Assembly che esegua la divisione tra 0003 FFFE : 2 = ….. Con il debug verificare il risultato: resto DX=00h e quoto in AX = FFFFh

```
.model small
.stack

.data

.code
 inizio:

   mov dx,0003h     ;dividendo in DX:AX
   mov ax,0FFFEh  ; se la cifra dell'esadecimale è una lettera farla precedere con 0
   mov bx,2         ;divisore
   div bx           ; in debug verificare il risultato: resto DX=00h e quoto in AX = FFFFh

   mov ah,4ch
   int 21h
 end inizio
```