

#01 Testo del problema

Dati due byte, calcolarne il prodotto.

#02 Analisi del problema

Partiamo col dire che nel processore 8086 esiste già un circuito elettronico per calcolare il prodotto di due parole a 8 e 16 bit, quindi in realtà questo programma sarà solo un esercizio di stile.

Esistono almeno due metodi per calcolare il prodotto di due byte: il primo, il più immediato da elaborare, è quello di utilizzare la definizione stessa del prodotto in matematica per creare il rispettivo programma:

“Dati due numeri interi A e B , si definisce prodotto tra A e B la somma di tanti addendi uguali ad A tante volte quanto è indicato da B ”.

Così, il prodotto tra 5 e 8, sarà uguale alla somma di otto addendi tutti uguali a cinque.

$$5 \cdot 8 = 5 + 5 + 5 + 5 + 5 + 5 + 5 + 5$$

Il secondo metodo è sicuramente meno immediato, ed è una conseguenza diretta del calcolo in colonna con cifre binarie.

Si osservi l'esempio sottostante:

$$\begin{array}{r}
 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ x \\
 1\ 0\ 1\ = \\
 \hline
 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0 \\
 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ - \\
 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ -\ - \\
 \hline
 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0
 \end{array}$$

Si possono ora fare alcune osservazioni:

- L'operazione finale è una somma di tanti addendi quante sono le cifre binarie del secondo fattore

- Partendo dal bit meno significativo del secondo fattore si aggiunge il primo fattore se il bit in esame è 1, nulla altrimenti
- Ad ogni cifra del secondo fattore analizzata, si moltiplica comunque l'attuale somma per 2 (si aggiunge uno zero a destra)

Questo secondo metodo ha il difetto di essere meno comprensibile, soprattutto se scritto in codice, ma ha il pregio di essere estremamente veloce.

Se si analizzano i costi in termini di tempo, si scopre che il primo programma esegue sempre B iterazioni, dove B è il numero del secondo fattore.

Il secondo, invece, esegue sempre $\log_2 B + 1$ iterazioni (approssimato per difetto), ossia un numero di iterazioni pari al numero di cifre necessarie alla rappresentazione binaria del secondo fattore.

C'è comunque da dire che in ognuna di queste iterazioni devono essere compiute tutte queste operazioni:

- Confronto del bit in esame con 0
- Raddoppio dell'attuale prodotto
- Eventuale somma del primo fattore
- Aggiornamento maschera

Decidiamo comunque di presentare entrambi i metodi, per mostrarne pregi e difetti.

#03 Progettazione delle routine

Nessuna routine necessaria.

#04 Registri utilizzati

Registri a 8 bit	
AH	▪ Primo fattore (metodo 1/2)
AL	▪ Secondo fattore (metodo 1/2)
DH	▪ Maschera (metodo 2)
Registri a 16 bit	
BX	▪ Prodotto (metodo 1/2)

#05 Codice del programma

p05.asm

```
-----  
;  
; Direttive all'assemblatore  
-----  
;  
DOSSEG          ; Segmentazione DOS  
.MODEL  small   ; Modello di memoria  
.STACK  100h    ; Dimensione dello stack  
  
-----  
; Data Segment  
-----  
;  
.DATA  
X      db      5  
Y      db      8  
P1     dw      ?  
P2     dw      ?  
  
-----  
; Code segment  
-----  
;  
.CODE  
  
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  
; Programma principale  
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  
Begin: StartupCode;          ; Inizio del programma  
  
;----- PRIMO METODO -----  
;-----  
; Prelevo i due fattori  
MOV  AH , [X]  
MOV  AL , [Y]  
; Inizializzo il prodotto a zero  
XOR  BX , BX  
  
Ciclo1: ; Verifichiamo se dobbiamo eseguire altre somme  
CMP  AL , 0  
JZ   FCiclo1  
; Aggiungiamo il primo fattore  
ADD  BL , AH  
ADC  BH , 0  
; Decrementiamo il secondo fattore  
DEC  AL  
; Ritorniamo alla condizione  
JMP  Ciclo1  
  
FCiclo1: ; Salvataggio prodotto  
MOV  [P1] , BX  
  
;----- SECONDO METODO -----  
;-----  
; Prelevo i due fattori  
MOV  AL , [X]  
XOR  AH , AH  
MOV  DL , [Y]  
; Inizializzo il prodotto a zero  
XOR  BX , BX  
; Inizializzo la maschera
```

```

MOV    DH , 1

Ciclo2: ; Vediamo se abbiamo finito i bit
        CMP    DL , 0
        JZ     FCiclo2
        ; Verifichiamo l'attuale bit
        TEST   DL , DH
        JZ     Dopo
        ; La cifra è 1, sommiamo il primo fattore
        ADD    BX , AX
        ; Alteriamo il fattore
        SUB    DL , DH
Dopo:   ; Aggiorniamo la maschera
        ADD    DH , DH
        ; Prossimo eventuale addendo
        ADD    AX , AX
        ; Ritorniamo alla condizione
        JMP    Ciclo2
FCiclo2: ; Salvataggio prodotto
        MOV    [P2] , BX

; Operazioni finali
ExitCode 0      ; Esecuzione corretta del programma
END Begin      ; Fine del programma

```

#06 Prove effettuate

Prova 01			
Significato della prova	▪ Dati pseudo – casuali		
Valori in input	X	=>	5
	Y	=>	8
Valori attesi in output	P1	=>	40
	P2	=>	40
Valori ottenuti in output	P1	=>	28h (40)
	P2	=>	28h (40)

Prova 02			
Significato della prova	▪ Dati pseudo – casuali		
Valori in input	X	=>	5
	Y	=>	5
Valori attesi in output	P1	=>	25
	P2	=>	25
Valori ottenuti in output	P1	=>	19h (25)
	P2	=>	19h (25)

Prova 03			
Significato della prova	▪ Primo operando nullo		
Valori in input	X	=>	0
	Y	=>	5
Valori attesi in output	P1	=>	0
	P2	=>	0
Valori ottenuti in output	P1	=>	0
	P2	=>	0

Prova 04			
Significato della prova	▪ Secondo operando nullo		
Valori in input	X	=>	5
	Y	=>	0
Valori attesi in output	P1	=>	0
	P2	=>	0
Valori ottenuti in output	P1	=>	0
	P2	=>	0

Prova 05			
Significato della prova	▪ Limite superiore		
Valori in input	X	=>	FFh
	Y	=>	FFh
Valori attesi in output	P1	=>	FE01h
	P2	=>	FE01h
Valori ottenuti in output	P1	=>	FE01h
	P2	=>	FE01h