

## #01 Testo del problema

---

Dati due byte, scrivere in memoria quanti di questi due numeri sono pari.

## #02 Analisi del problema

---

Il metodo risolutivo per questo problema è abbastanza semplice ed immediato: uno alla volta, si analizzano i due byte in questione, incrementando ogni volta un contatore nel caso in cui il numero analizzato sia pari. Per verificare che un numero sia pari è sufficiente controllare l'ultima cifra. Nella configurazione binaria, infatti, tutti i bit più significativi, ad eccezione dell'ultimo, hanno un valore pari (2, 4, 8, 16, 32 e così via ogni volta moltiplicando per 2), mentre l'ultimo bit ha valore dispari (1). Sommando una cifra dispari (1) ad una certamente pari (tutte le altre), si ottiene sicuramente una cifra dispari.

Possiamo quindi concludere che, se l'ultimo bit di una cifra binaria è in stato logico 1, la cifra è dispari, pari altrimenti.

Per controllare l'ultimo bit utilizziamo la tecnica del mascheramento. Azzeriamo tutti i bit escluso l'ultimo, che lasciamo invariato: se il risultato di questa operazione è zero, allora la cifra è pari, altrimenti è dispari.

L'operazione che ci permette di eseguire il tutto è la congiunzione logica con il byte 00000001. I primi 7 bit sono forzati a zero, l'ultimo rimane invece invariato.

Ad esempio, il numero 10 (1010b) è così mascherato:

```
0 0 0 0 1 0 1 0 &
0 0 0 0 0 0 0 1 =
0 0 0 0 0 0 0 0 → Risultato uguale a zero, numero pari.
```

Con il numero 15 (1111b) si ottiene:

```
0 0 0 0 1 1 1 1 &
0 0 0 0 0 0 0 1 =
0 0 0 0 0 0 0 1 → Risultato diverso da zero, numero dispari.
```

Da sottolineare che, come in quasi tutte le computazioni, lo zero è considerato numero pari.

## #03 Progettazione delle routine

---

pari		
Compiti svolti	▪ Verifica se un byte è pari	
Parametri di input	AL	Byte da analizzare
Parametri di output	CY	1 => Cifra pari 0 => Cifra dispari
Risorse necessarie	Nessuna risorsa necessaria	
Registri sporcati	Nessun registro sporcato	
Limiti della routine	Nessun limite di utilizzo	
Creatore della routine	Andrea Asta ( <a href="mailto:asta.andrea@tin.it">asta.andrea@tin.it</a> )	

## #04 Registri utilizzati

---

Registri a 8 bit	
AL	▪ Numero in analisi
CH	▪ Utilizzo del registro
Registri a 16 bit	
Nessun registro a 16 bit utilizzato	

## #05 Codice del programma

---

```
p03.asm
;-----
; Directive all'assemblatore
;-----
DOSSEG                ; Segmentazione DOS
.MODEL                 small      ; Modello di memoria
.STACK                100h       ; Dimensione dello stack

;-----
; Data Segment
;-----
.DATA
X      db      100
Y      db      101
N      db      ?

;-----
; Code segment
;-----
.CODE

;-----
; PROCEDURA: pari
; Verifica se un byte è pari
; INP: AL
; OUT: CF (1 pari, 0 dispari)
;-----
```

```

pari PROC
    PUSHF                ; Salvataggio flags
    TEST    AL , 1       ; Eseguo l'intersezione logica
    JZ      pari1        ; Se salto, il numero è pari
    POPF                ; Recupero il registro dei flag
    CLC                ; Azzero il Carry
    JMP     pari2        ; Vado alla fine della routine
pari1: POPF                ; Recupero il registro dei flag
    STC                ; Setto il Carry a 1
pari2: RET                ; Torno al chiamante
    ENDP                ; Fine della procedura

; Programma principale
Begin: StartupCode      ; Inizio del programma

; Inizializzazione variabili
    XOR    CH , CH      ; Azzero il contatore
    MOV    AL , [X]     ; Prelevo il primo numero

; Algoritmo
    CALL   pari         ; Verifico se è pari
    JNC   nNum         ; Se è dispari, passo al prossimo numero...
    INC   CH           ; ... altrimenti incremento il contatore

nNum: ; Ripetiamo le stesse operazioni con l'altro numero
    MOV   AL , [Y]     ; Prelevo il secondo numero
    CALL  pari         ; Verifico se è pari
    JNC  salva        ; Se salto, è dispari
    INC  CH           ; ...altrimenti incremento il contatore

; Salvataggio risultati
salva: MOV    [N] , CH ; Salvo il contatore

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

```

## #06 Prove effettuate

Prova 01		
Significato della prova	▪ Un numero pari e uno dispari	
Valori in input	X	=> 100
	Y	=> 101
Valori attesi in output	N	=> 01
Valori ottenuti in output	N	=> 01

Prova 02			
Significato della prova	▪ Entrambi i numeri pari		
Valori in input	X	=>	100
	Y	=>	100
Valori attesi in output	N	=>	02
Valori ottenuti in output	N	=>	02

Prova 03			
Significato della prova	▪ Entrambi i numeri dispari		
Valori in input	X	=>	101
	Y	=>	101
Valori attesi in output	N	=>	00
Valori ottenuti in output	N	=>	00

Prova 04			
Significato della prova	▪ Entrambi i numeri nulli		
Valori in input	X	=>	00
	Y	=>	00
Valori attesi in output	N	=>	02
Valori ottenuti in output	N	=>	02