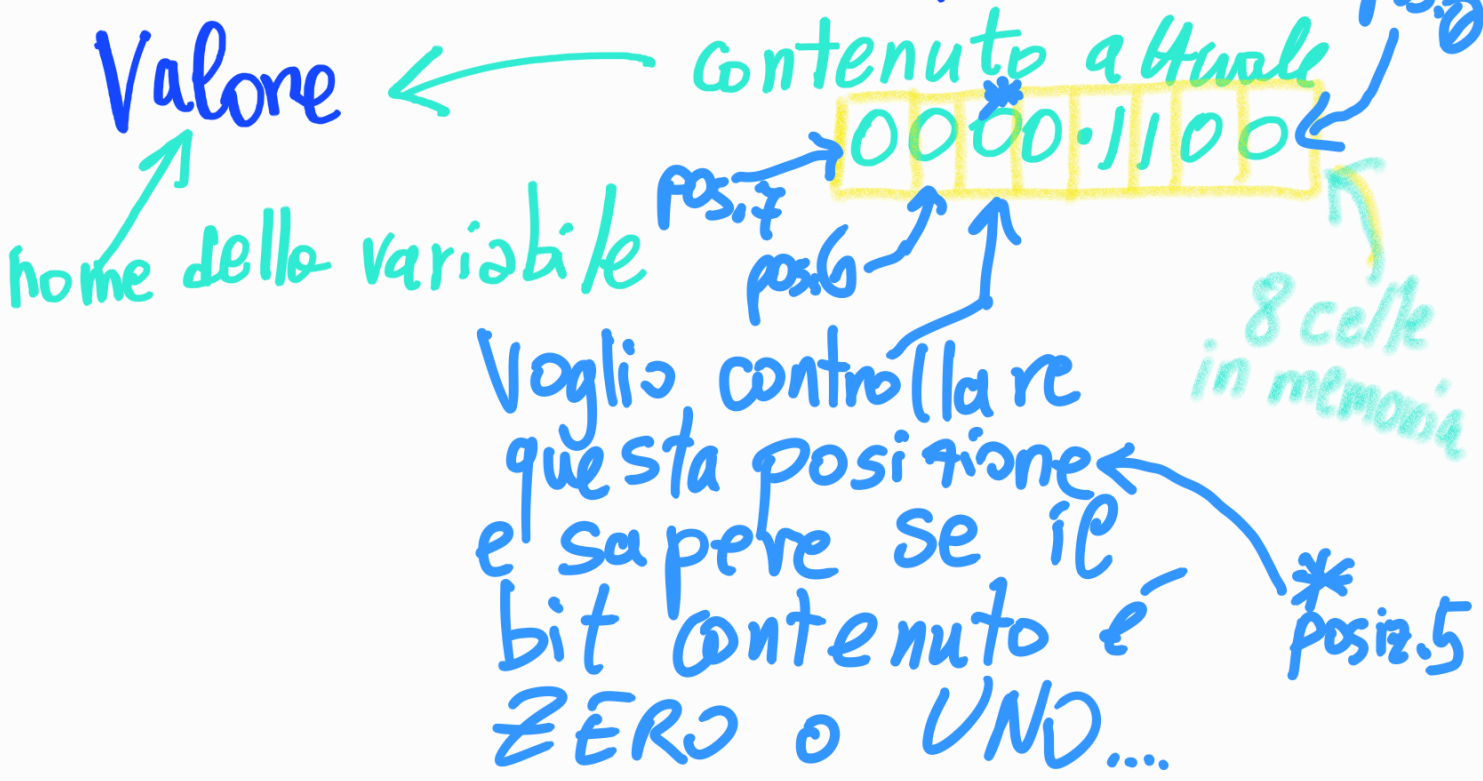
DI NUOVO SULLE MASCHERATURE...

le mascherature si usano per...

1) controllare uno o più bit "ignorando" tutti gli altri
2) modificare uno o più bit lasciando inalterati tutti gli altri

CONTROLLO DI BIT

Voglio verificare se un bit (in un numero a 8 bit) è a 1 oppure a 0

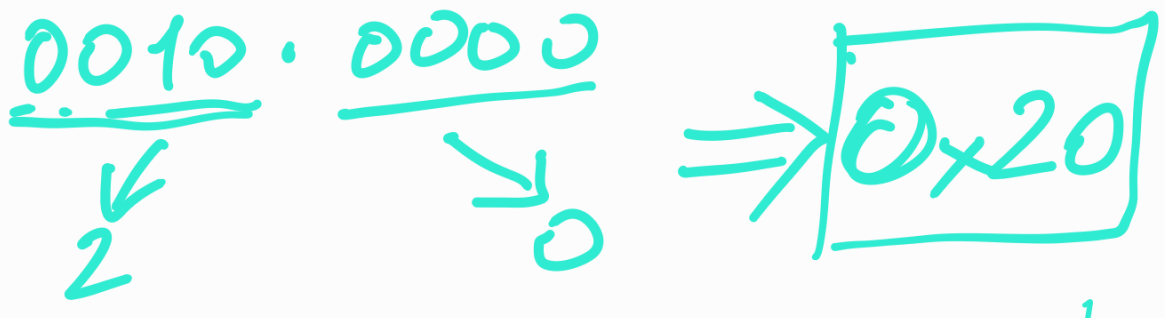


Preparo la maschera adeguata per effettuare il controllo....

Maschera = 0010·0000

↖ solo il bit in posizione 5 è a 1, tutti gli altri sono a zero...

Scriviamo il numero in esadecimale



Ecco il controllo:

maschera

```
if ( Valore & 0x20 )
{
  // Se bit era 1...
}
else
{
  // Se bit era 0...
}
```

Valore da verificare

AND

il risultato può essere ZERO (falso) oppure NON-ZERO (true)...

Entriamo un altimo nei dettagli:

L'operazione effettuata dentro la funzione if() è una operazione matematica binaria (AND) tra due numeri: la variabile Valore e la cifra 0x20.
Rivediamola in cifre binarie:

bit che voglio controllare

Valore → 0 0 0 0 · 1 1 0 0

AND

Maschera → 0 0 1 0 · 0 0 0 0 ← 0x20

Risultato → 0 0 0 0 · 0 0 0 0 ← 0x00

Se risultato numerico è ZERO che, nella funzione if() viene interpretato come VERO (true).

Ricordiamo che le tavole della verità di AND, OR e XOR sono queste:

AND

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

OR

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

XOR

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

C'è anche da notare che l'operazione viene eseguita bit a bit ovvero una singola cifra binaria alla volta...

Il controllo si può effettuare anche con più di un bit per volta.

Ad esempio:

- Voglio controllare se uno qualsiasi dei 3 bit alti di una cifra a 16 bit è ad UNO....

voglio controllare questi 3 bit

Valore → 0100 · 0110 · 1100 · 1101

Maschera → 1110 · 0000 · 0000 · 0000 ← 0xE000

```
if (Valore & 0xE000)
```

Sono SEMPRE due soli risultati possibili:

- ZERO (interpretato come FALSE) ovvero nessuno dei tre bit mascherati

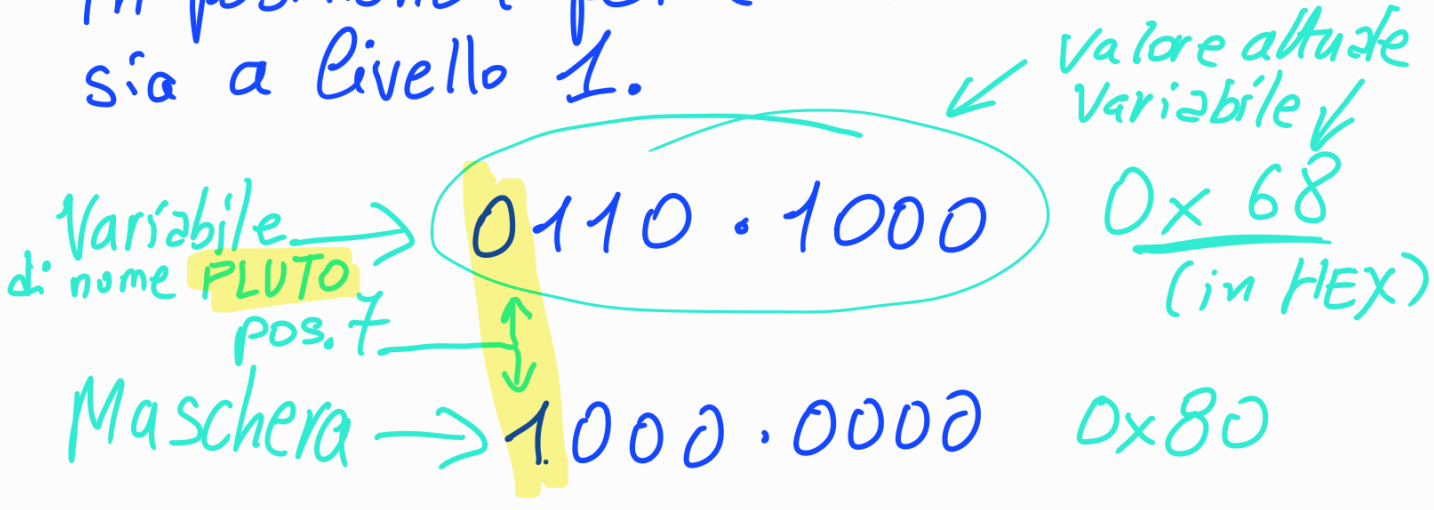
era ad UNO

- NON_ZERO (interpretato come TRUE) ovvero almeno uno dei tre bit mascherati era ad UNO.

MODIFICA DI BIT

L'altra possibilità offerta dagli operators AND, OR e XOR è quella di modificare lo stato di UN BIT (o alcuni), senza modificare gli altri...

Ad esempio: in un numero a 8 bit voglio modificare lo stato del bit in posizione 7 per essere sicuro che sia a livello 1.



Volendo imporre il valore UNO su di un bit occorre utilizzare l'operazione algebrica binaria OR.

```
PLUTO = PLUTO | 0x80;
```

- PLUTO (a sinistra): nuovo valore = 0xE8
- =: scrittura del nuovo valore
- PLUTO (a destra): vecchio valore 0x68
- |: operazione OR
- 0x80: maschera

questo perché

PLUTO (vecchio valore) →	0110 · 1000	0x68
	↓↓↓↓OR↓↓↓↓	
maschera →	1000 · 0000	0x80
	1110 · 1000	0xE8
	E 8	

Si può effettuare la modifica su più bit contemporaneamente...

Voglio portare TUTTI i 4 bit alti dell'ottetto (BYTE) a livello 1.

PLUTO (vecchio valore) → 0110 . 1000 0x68

OR

maschera → 1111 . 0000 0xF0

PLUTO (nuovo valore) → 1111 . 1000 0xF8

Se invece voglio forzare uno o più bit di un numero al valore ZERO devo necessariamente utilizzare la funzione algebrica binaria AND, con l'accortezza però di usare una mascheratura "invertita" rispetto agli esempi utilizzati finora.
Questo è necessario poiché nella tabella della verità dell'operazione AND ha la priorità lo ZERO...

Esempio:

in un numero a 16 bit voglio porre tutti gli 8 bit alti a ZERO