Gli operatori bit a bit

- Lavorano sugli interi e i caratteri con segno e senza segno
- & (and), | (or), ^ (xor), ~ (complemento)
  - Lavorano sui bit corrispondenti dei valori coinvolti

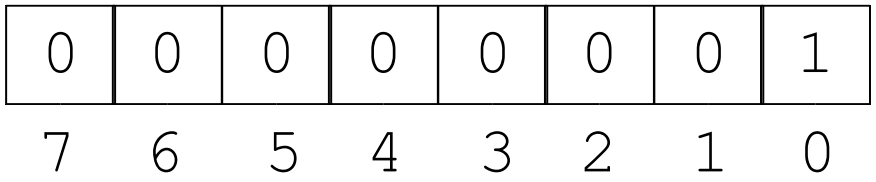
Gli operatori bit a bit

- Lavorano sugli interi e i caratteri con segno e senza segno
- << (lshift), >> (rshift)
  - Spostano verso destra o verso sinistra la rappresentazione binaria ad esempio
  - `int a = 1, b = 567;`

```
a =  a <<  3; /* a vale 1000 */
b = b & ~ (1 <<  4 )
/* azzera il quinto bit di b ...
Quanto vale ora b? */
```

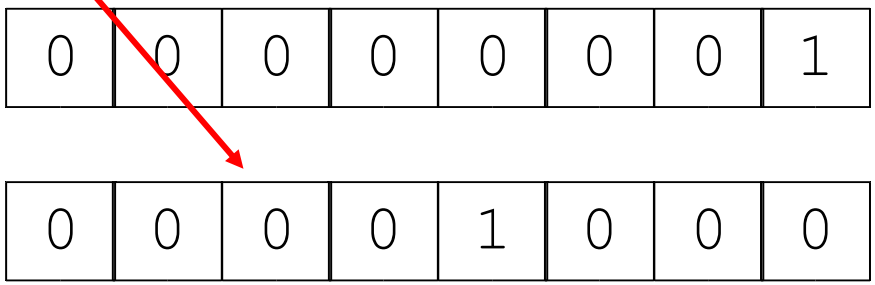
Es operatori bit-a-bit

- a = a & ~(1 << 3)



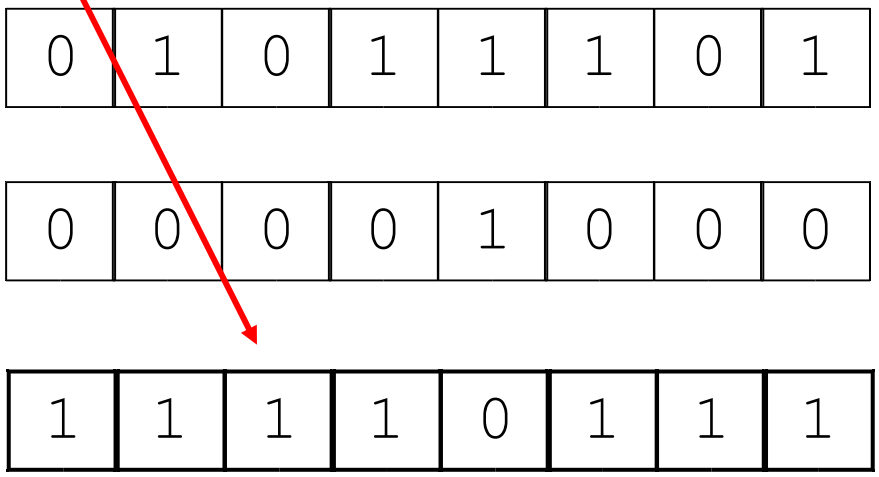
Es operatori bit-a-bit (2)

- a = a & ~(1 << 3)



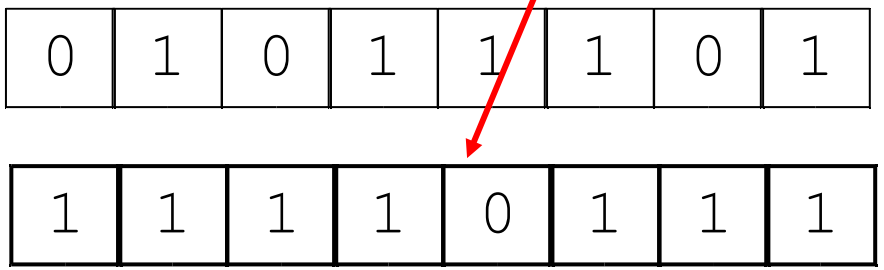
Es operatori bit-a-bit (3)

- a = a & ~ (1 << 3)



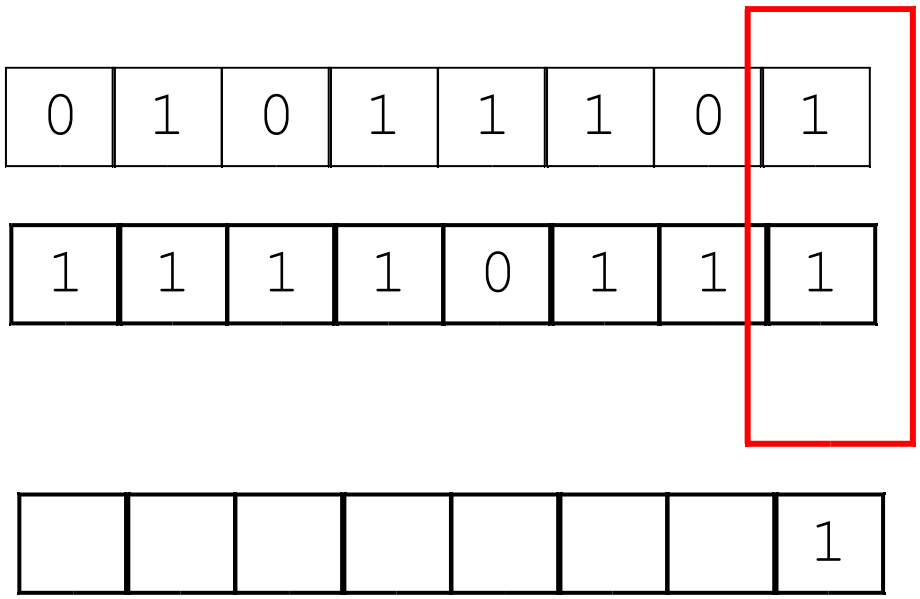
Es operatori bit-a-bit (4)

- a = a & ~(1 << 3);



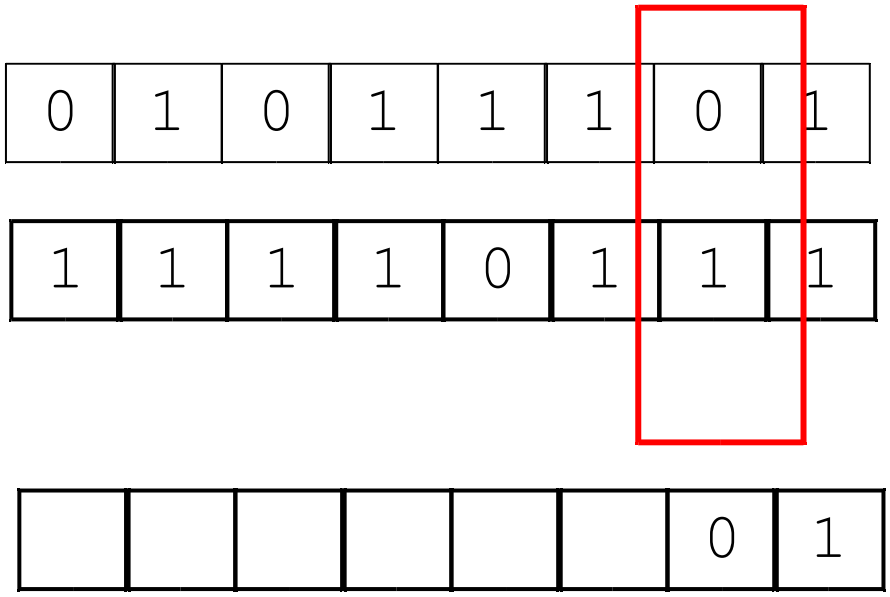
Es operatori bit-a-bit (5)

- `a = a & ~(1 << 3 )`



Es operatori bit-a-bit (6)

- `a = a & ~ (1 << 3 )`



Es operatori bit-a-bit (7)

- `a = a & ~ (1 << 3 )`

0	1	0	1	1	1	0	1

1	1	1	1	0	1	1	1

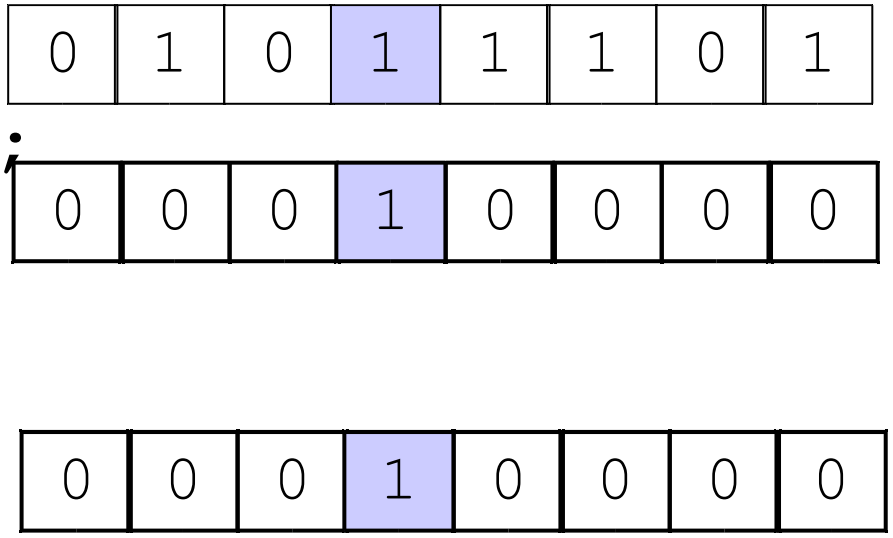
0	1	0	1	0	1	0	1

`Nuovo valore di a`

Abbiamo azzerato il bit di posizione `3`

Selezionare l'n-esimo bit di un intero

```
int c=456, int n=4, int i;
int bit_i;
if (((1<<n) & c)!= 0)
        bit_i = 1;
    else
        bit_i = 0;
```



Stampare i K bit meno significativi di un intero

```
int c = 567, i;


for (i=0; i<K; i++)
  if (((1<<i) & c)!= 0)
        printf(”1”);
    else
        printf(”0”);
```