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

A_i	B_i	A | B	~A	A&B	A^B
0	0	0	1	0	0
0	1	1	1	0	1
1	0	1	0	0	1
1	1	1	0	1	0

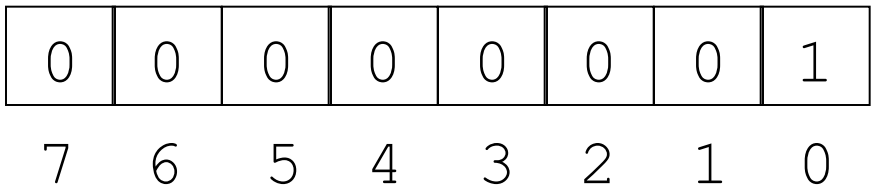
Gli operatori bit a bit

- 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 = 93, b = 1
 b =  b <<  3; /* b vale 1000 (complemento
a 2)*/
 a = a & ~ b;
/* azzera il quarto bit di a ...
Quanto vale ora a? */
```

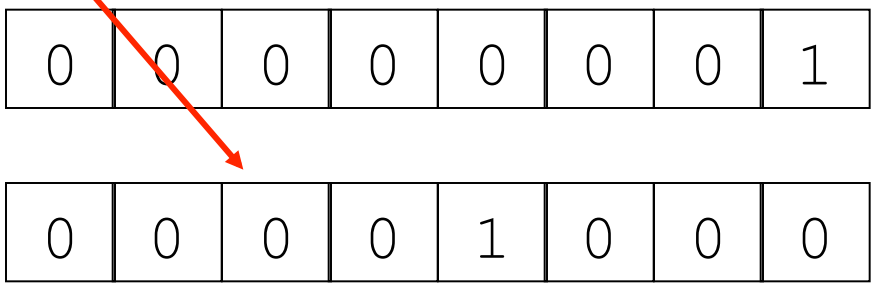
Es operatori bit-a-bit

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



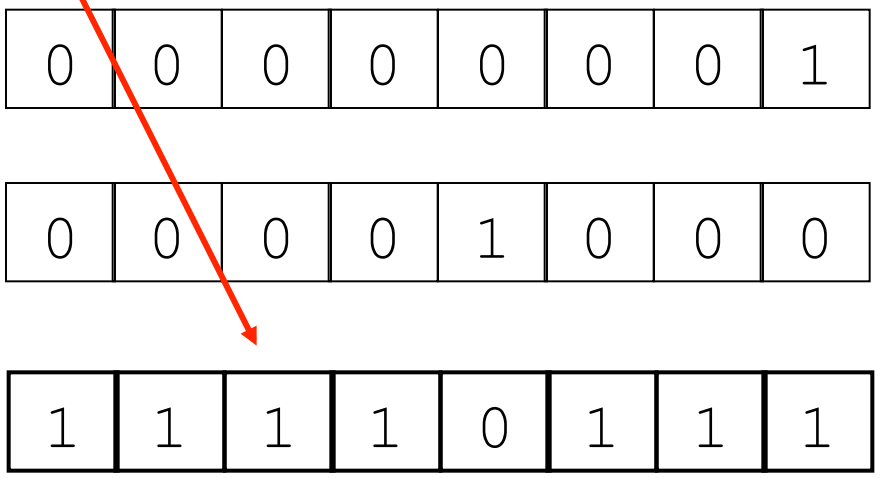
Es operatori bit-a-bit

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



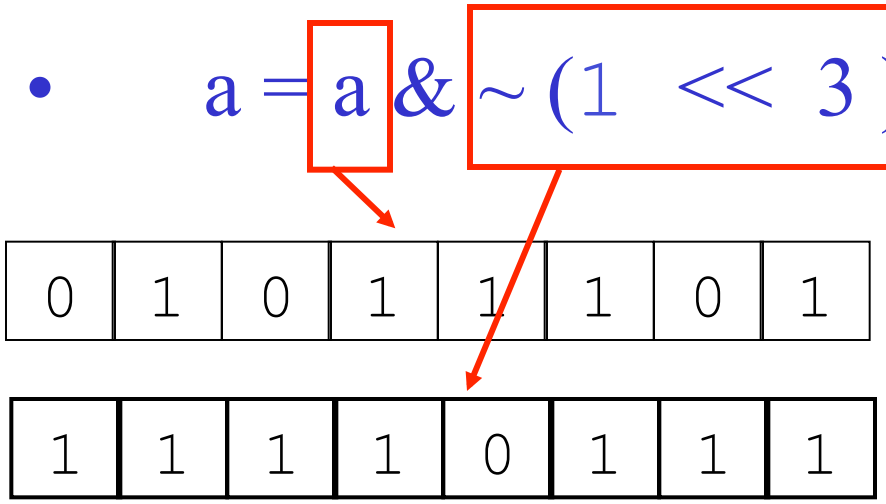
Es operatori bit-a-bit

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



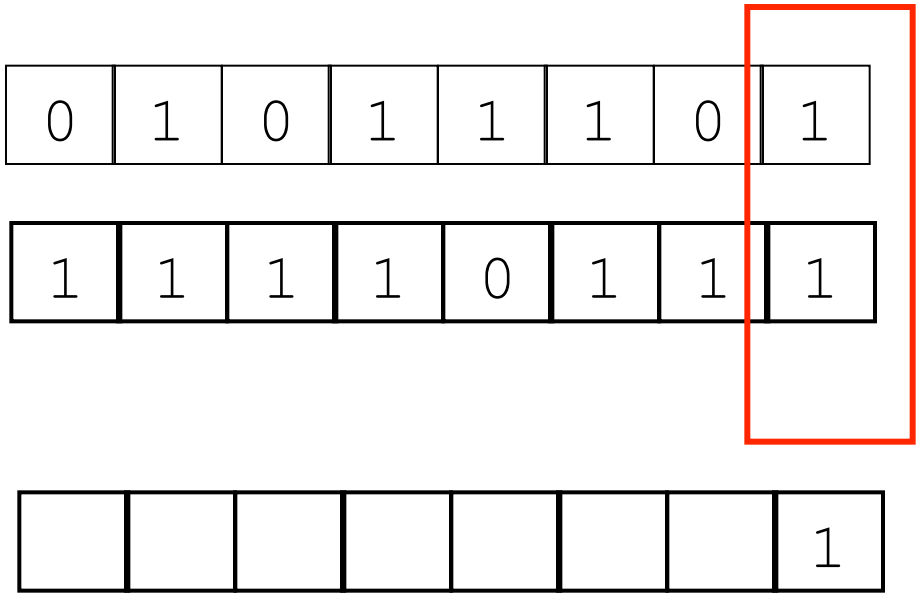
Es operatori bit-a-bit

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



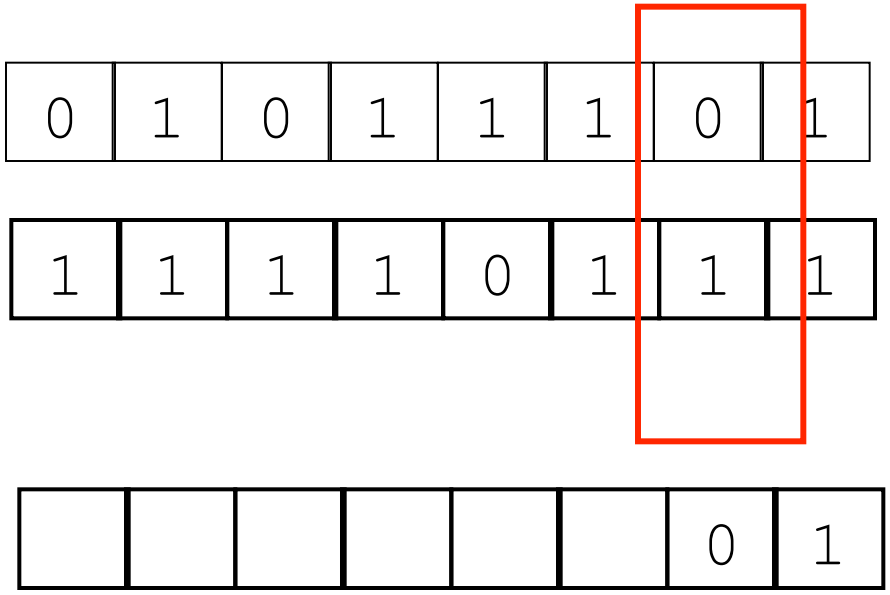
Es operatori bit-a-bit

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



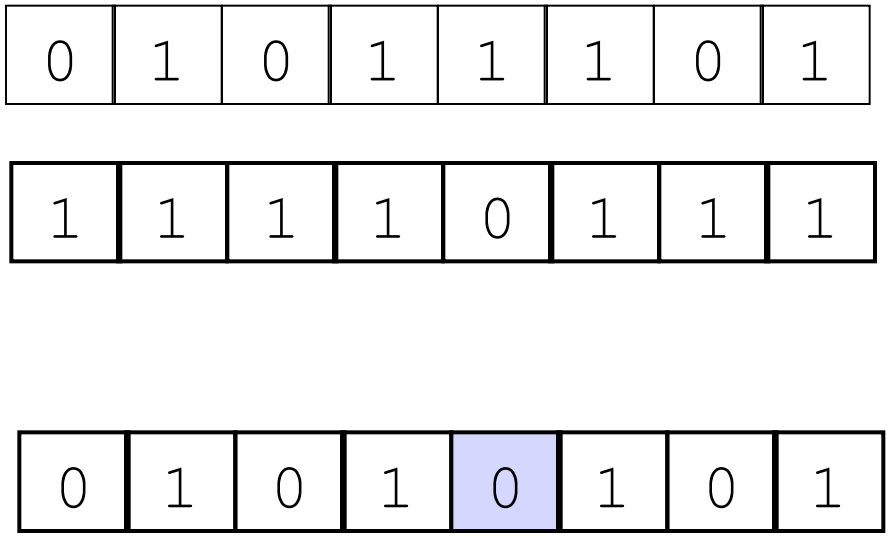
Es operatori bit-a-bit

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



Es operatori bit-a-bit

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

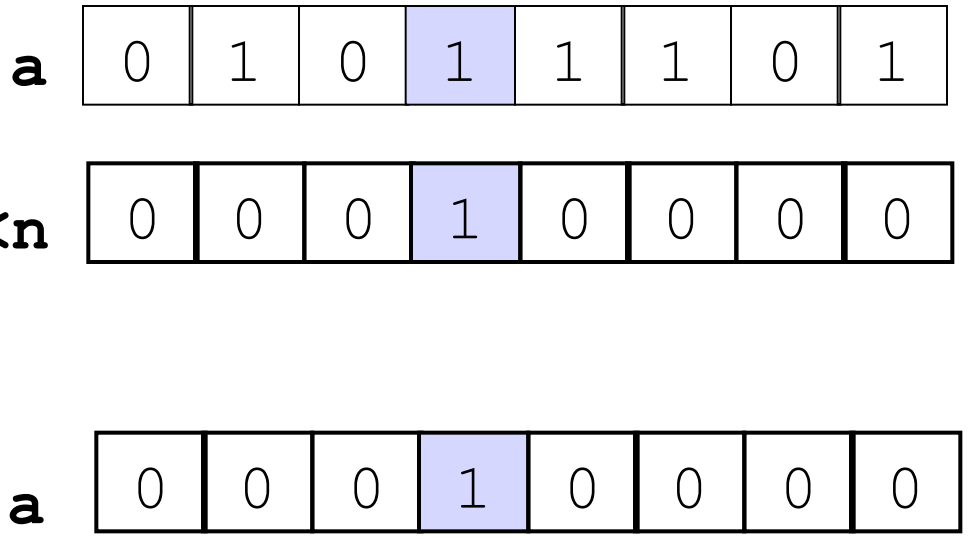



```
Nuovo valore di a =85
```

Abbiamo azzerato il bit `3`

Selezionare l'n-esimo bit di un intero

```
int a=93; int n=4; int i;
int bit_n;
if (((1<<n) & a)!= 0)
     bit_n = 1;
  else
     bit_n = 0;
```



Stampare i K bit meno significativi di un intero

```
int a = 93, i;

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