Codifica binaria

La codifica alla base di ogni sistema per il trattamento dell'informazione:

11001 → 2 ^4 * 1 + 2^3 * 1 + 2^2 * 0 + 2^1 * 0 + 2^0 * 1 = 25

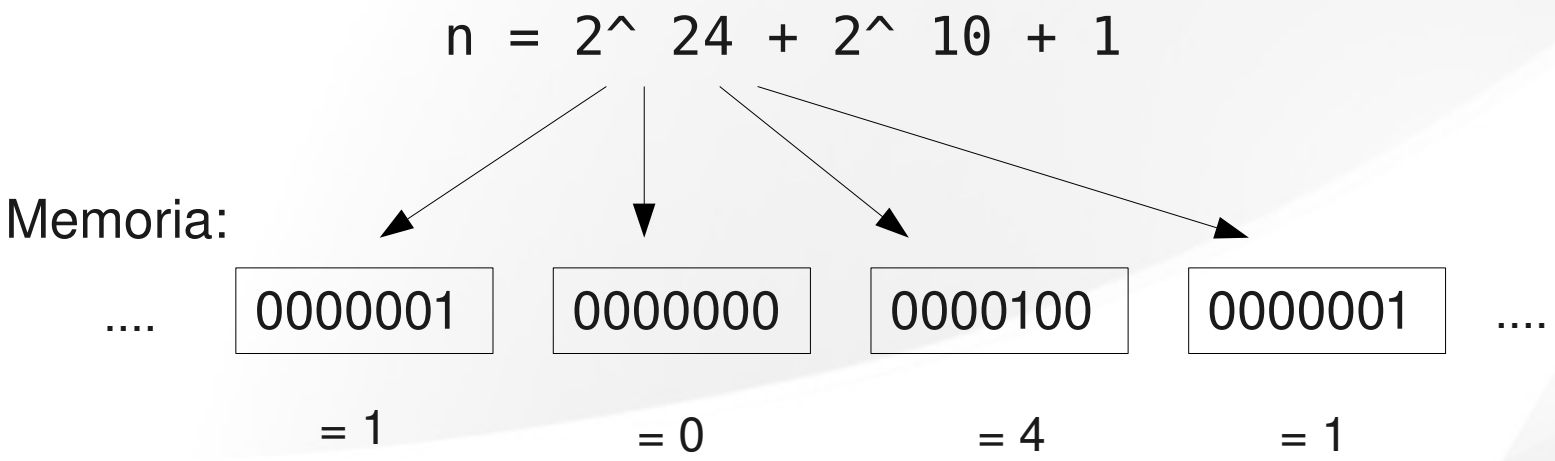
L'unità di memoria di base è il byte, che contiene 8 bits di informazione:

N	bits
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110

Codifica dei tipi di dato

Ogni tipo di dato è rappresentato in memoria da una concatenazione di bytes:

```
unsigned int |  4 bytes
```



Architettura di tipo Big-Endian

Codifica dei tipi di dato

Le operazioni aritmetiche restituiscono il valore modulo 2^w dove w è l'ampiezza in bit del tipo di dato:

255 + 1 == 0 nell'aritmetica dei char (w = 8)

Questo permette di:

- Gestire gli overflow nei risultati delle operazioni
- Rappresentare in maniera comoda gli interi con il segno

Es.: 255 == -1 tra i char con segno

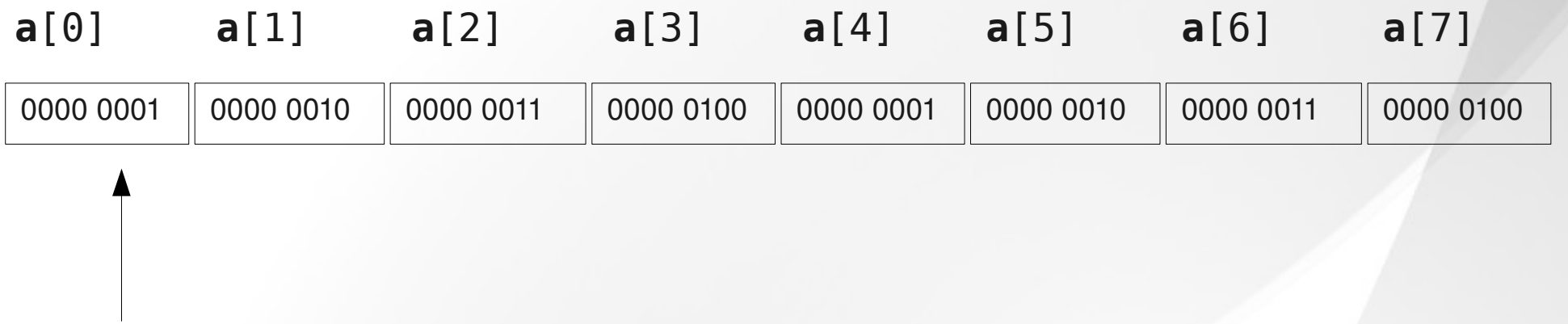
In generale: -x = 2^w – x (regola del complemento a 2)

Esercizio 4

Scrivere un programma che, ricevuto in input un intero n, genera un array A di 4*n unsigned char (quindi interi in [0,255]) riempito con elementi casuali.

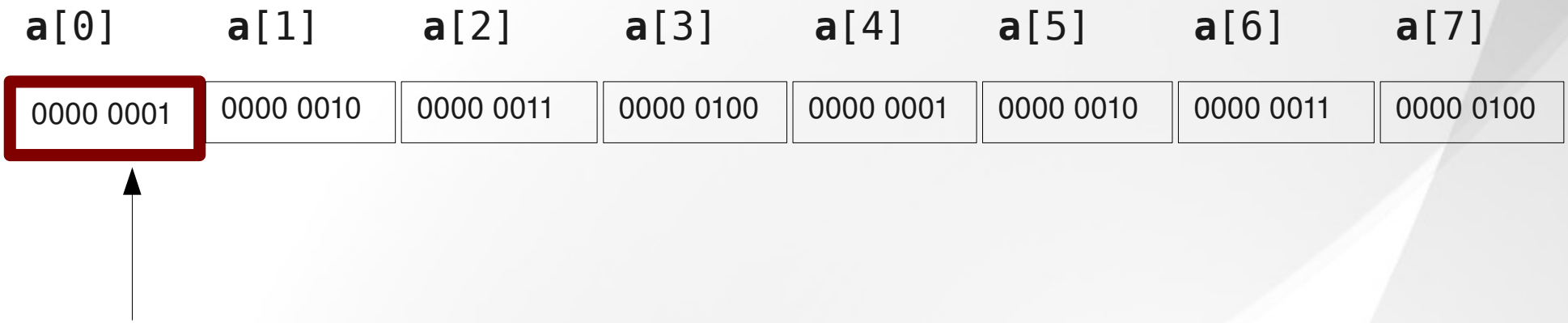
Quindi calcola il massimo dei valori contenuti in A nell'ipotesi di interpretare il suo contenuto come array di 4*n char, 2*n short, e n int, e stampa i tre risultati così ottenuti.

Layout in memoria



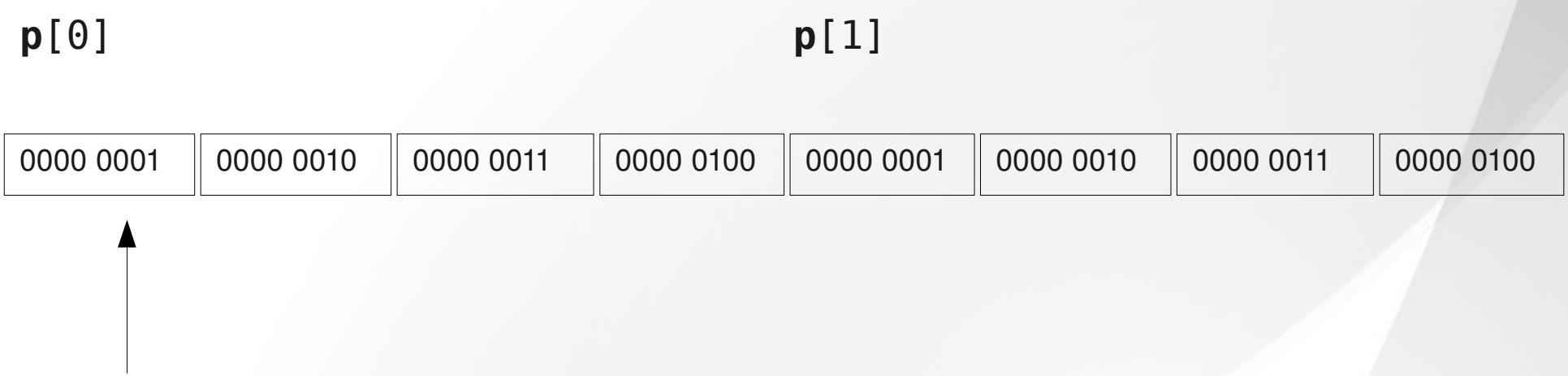

```
unsigned char *a;
```

Layout in memoria




```
unsigned char *a;     *a (o a[0]) restituisce  1
```

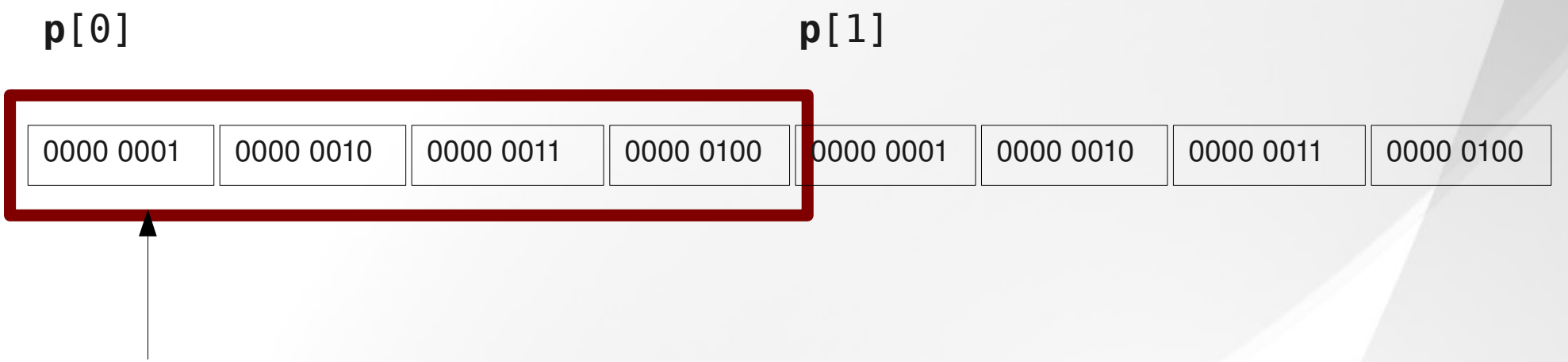
Layout in memoria




```
unsigned char *a;     *a (o a[0]) restituisce  1
```

```
int *p = (int*)a;
```

Layout in memoria




```
unsigned char *a;
```

*a (o a[0]) restituisce 1

```
int *p = (int*)a;
```

*p (o p[0]) restituisce:

bin 0000 0001 0000 0010 0000 0011 0000 0100

dec 16909060