Rappresentazione binaria

Size | Small topic

Rappresentazione binaria

Rappresentazione binaria

Da base 10 a base 2:

154	QUOZIENTE	RESTO
/ 2	77	0
/ 2	38	1
/ 2	19	0
/ 2	9	1
/ 2	4	1
/ 2	2	0
/ 2	1	0
/ 2	0	1

154 *base 10* = **0011010** *base 2*

Funzione che prende un numero naturale in base 10 e restituisce le cifre in base 2

Rappresentazione dell’informazione

I computer usano la rappresentazione binaria, invece i circuiti elettronici la possono rappresentare e fare operazioni facilmente (e a costo basso) ON/OFF, porte logiche

- **Bit**: *unità elementare di informazione, ha (solo) due stati — il minimo perché si possa parlare di cose diverse. Convenzionalmente, li denotiamo con 0 e 1*
- ***Byte*** *: 8 bit, unità elementare di memoria, 28 valori distinti*
- ***Parola di memoria***: *8, 16, 32, 64, … bit dimensione dei registri nella CPU.*

Funzioni

Funzione che stampa le cifre di un n

```
function cifre(n) {
  let resto;
  while(n>0) {
    resto = n%10
    console.log(String(resto))
    n = (n-resto)/10
  }
}
```

Funzione che stampa le cifre di un numero, dalla più significativa alla meno significativa

```
function cifreRic(n) {
  if(n==0)
    return
  var resto = n%10
  cifreRic((n-resto)/10)
  console.log(resto)
}
```

Funzione che stampa le cifre di un numero (rappresentato in base 10) in altra base "b" (b<10)

```
function inbase(n,b) {
  let resto
  let codifica = ""
  while(n!=0) {
    resto = n%b
    n = (n-resto)/b
    codifica = String(resto) + codifica
  }
  console.log(codifica)
}


function inbaseRic(n,b) {
  if(n!=0) {
    let resto = n%b
    inbase((n-resto)/b,b)
    console.log(resto)
  }
}
```

Funzione che dati due interi a e b, rimuovere da a i bit settati a 1 in b

```
function rimuoviBit(a,b) {
  console.log("a")
  inbase(a,2)
  console.log("b")
  inbase(b,2)
  a = a & (~b)
  console.log("a")
  inbase(a,2)
}
```

Funzione che conta i bit settati a 1 in un numero a

```
function ones(a) {
  let ones = 0
  while (a!=0) {
    ones += a&1 // 1 se ultimo bit a destra è 1, 0 altrimenti
```

```
    a = a >>> 1 // butto via ultimo bit, da sx entra 0
  }
  return ones
}

console.log("ones(4): " + ones(4)) // 4_10 = 100_2
console.log("ones(7): " + ones(7)) // 7_10 = 111_2
console.log("* * *")
```

Funzione che scambia i valori di due variabili a e b utilizzando lo XOR

```
let x = 1
let y = 2
console.log("original -> x: " + x + " - y: " + y)
// SWAP
x = x^y
y = y^x // y^(x^y) = x
x = x^y
console.log("swapped -> x: " + x + " - y: " + y)
console.log("* * *")
```

Ricerca lineare - funzione che cerca un elemento in un array e restituisce la posizione della prima sua occorrenza nell'array (-1 se non c'è)

```
function search(e, a) {
  for (let i=0; i<a.length; i++)
    if (a[i] == e) return i
  return -1
}

console.log("search(10,[4,3,6,10,5,10]): " + search(10, [4, 3, 6, 10, 5, 10]))
console.log("search(15,[4,3,6,10,5,10]): " + search(15, [4, 3, 6, 10, 5, 10]))
console.log("* * *")
```

Ricerca binaria (*iterativa*) - funzione che cerca un elemento in un array ordinato e restituisce la sua posizione (-1 se non c'è)

```
function binarySearch(e, a) {
  let left = 0
  let right = a.length-1
  let mid
  while (left <= right) {
    mid = Math.floor((left+right)/2)
    if (e == a[mid]) return mid
    if (e > a[mid]) left = mid + 1
    if (e < a[mid]) right = mid - 1
  }
  return -1
}

console.log("binarySearch(10,[3,4,5,6,10]): " + binarySearch(10, [3, 4, 5, 6, 10]))
console.log("binarySearch(15,[3,4,5,6,10]):" + binarySearch(15, [3, 4, 5, 6, 10]))
```