TIPI PLOATING POINT

float 32

float64

$\mathbb{R}$

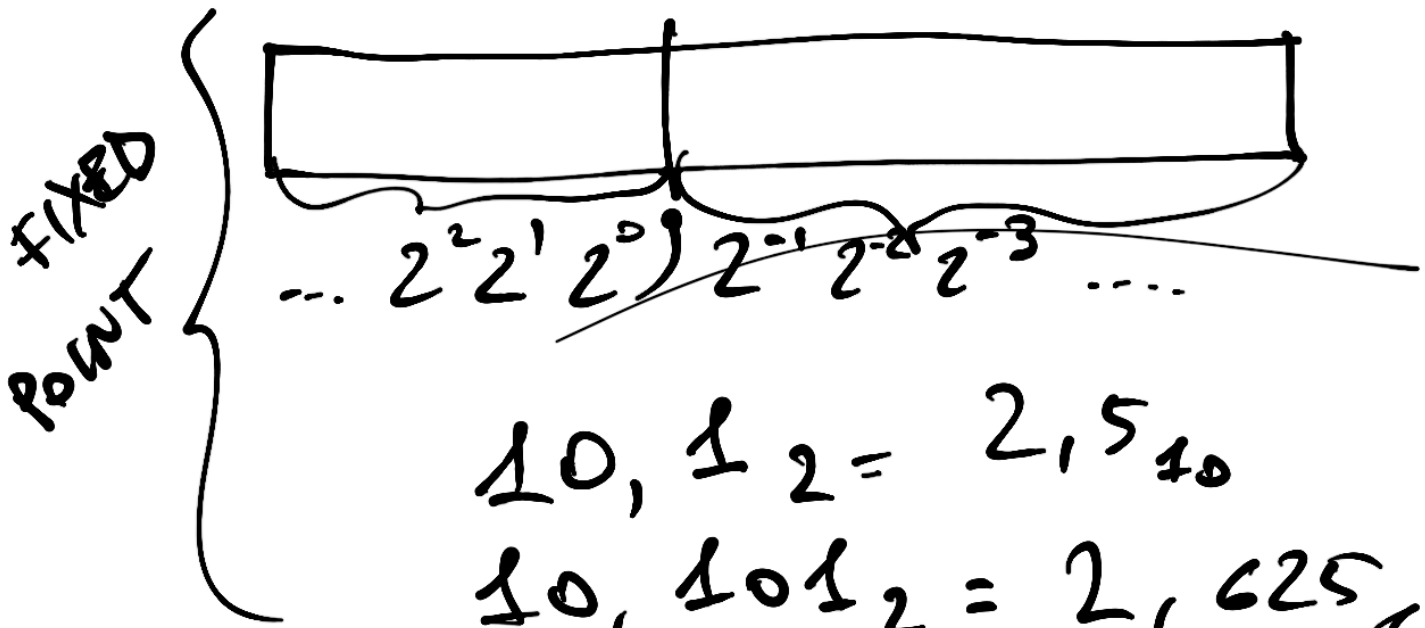
complex64

complex128

$\mathbb{C}$

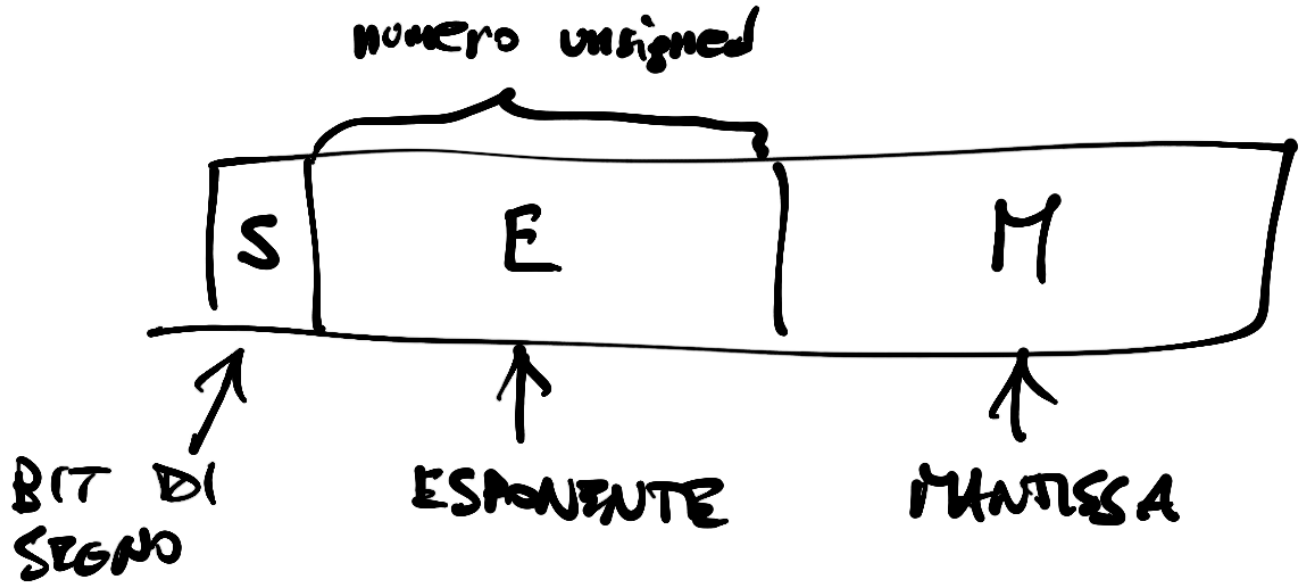
$a+ib$

- STANDARD IEEE 754



$10,1_2 = 2,5_{10}$

$10,101_2 = 2,625_{10}$



$$(-1)^S \; 2^{E-bias} \; \underbrace{1,M}$$

	#bit E	#bit M	bias
float32	8	23	127
float64	11	52	1023

	RANGE	PRECIS. decimale
float32	$\pm 10^{-45} \longrightarrow \pm 3{,}4 \cdot 10^{38}$	7 cifre
float64	$\pm 5 \cdot 10^{-324} \longrightarrow \pm 1{,}7 \cdot 10^{308}$	15 cifre

$\frac{1}{10} = 0.1_{10}$

$\frac{1}{3} = 0.333333\ldots = 0.\overline{3}$

float64 x

$x = 0.1$

0.1000000014901161...

FRA FLOAT NON E' P.

== !=

if (espr1 == espr2) { | NO

- -

}

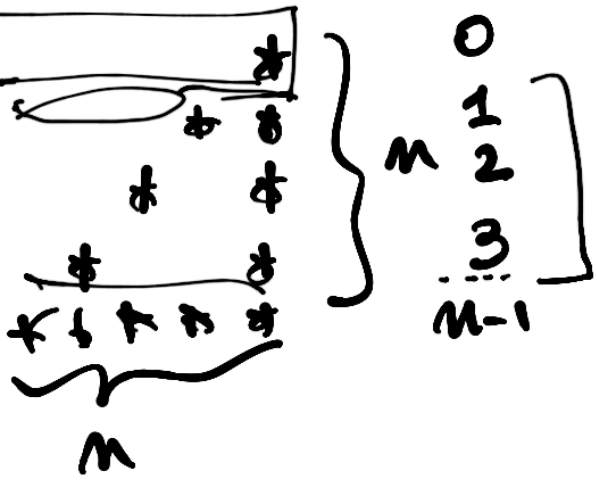
if Math.Abs(esp1 - esp2) < 1E-10 {

- - -

}

LETTERALI F.P.

3.17	3.17
3.	3.0
.14	0.14
0.14	0.14
1.12E-4	$1.12 \cdot 10^{-4}$



SP, *, SP, *, ↵

r
1 → n-2
2 n-3

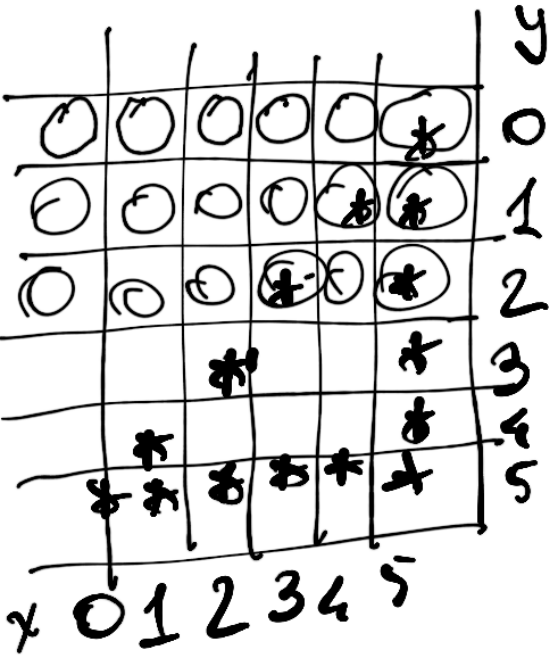
```
for i := 0; i < n-1; i++ {
        fout.Print(" ")
}
fout.Println("*")
```

1ª RIGA

```
for r := 1; r <= n-2; r++ {
    for i := 0; i < n-r-1; i++ {
        fout.Print(" ")
    }
    fout.Print("*")
    for i := 0; i < r-1; i++ {
        fout.Print(" ")
    }
}
```

```
for i := 0; i < n; i++ {
        fout.Print("*")
}
fout.Println()
```

ULTIMA RIGA




```
for y := 0; y < n; y++ {
    for x := 0; x < n; x++ {
        if x == n-1 || y == n-1 || x+y == n-1 {
            fmt.Print("*")
        } else {
            fmt.Print(" ")
        }
    }
    fmt.Println()
}
```

SEQUENZE DI COLLATZ

$$x \mapsto \begin{cases} x/2 & \text{se } x \text{ è pari} \\ 3x+1 & \text{se } x \text{ è dispari.} \end{cases}$$

$3 \to 10 \to 5 \to 16 \to$
$\to 8 \to 4 \to 2 \to (1)$

$13 \to 40 \to 20 \to 10 \to 5$
$\to 16 \to 8 \to 4 \to 2 \to (1)$

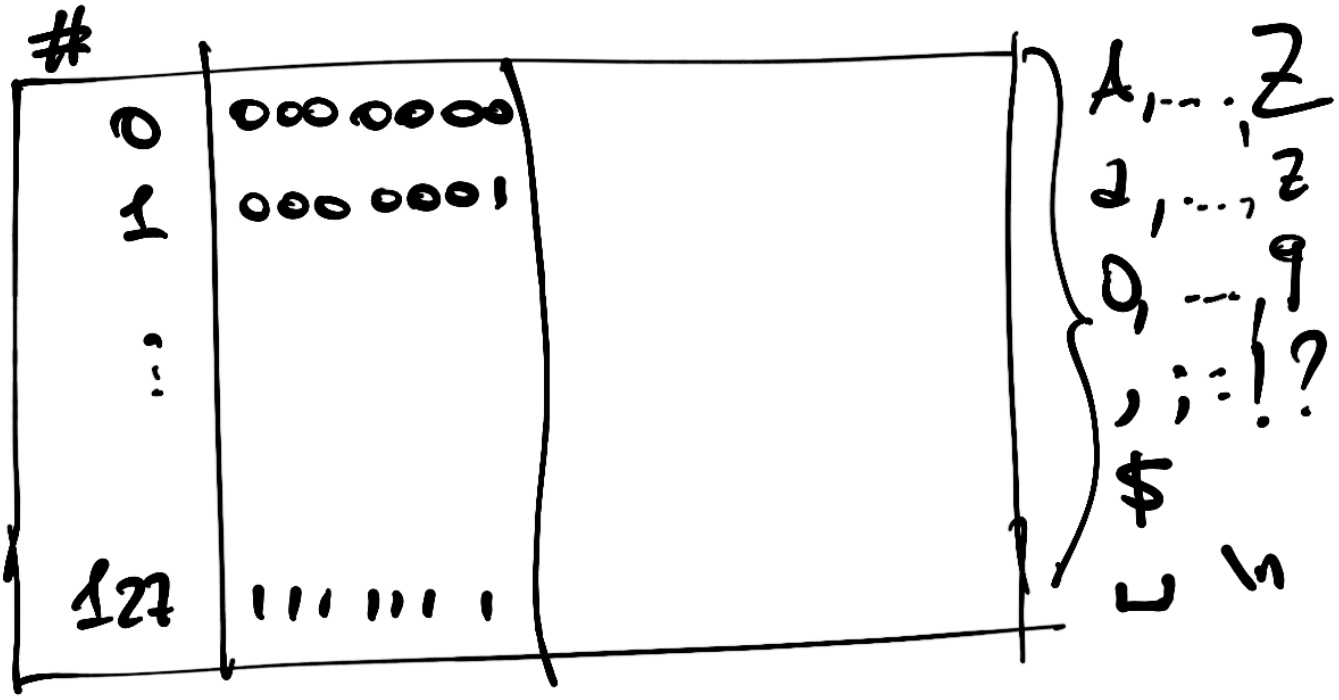
$17 \to 52 \to 26 \to 13 \to 40 \to$
$\to 20 \to 10 \to 5 \to 16$
$\to 8 \to 4 \to 2 \to (1)$

```
var x int
fmt.Scan(&x)
for {
    fmt.Println(x)
    if x==1 {
        break
    }
    if x%2 ==0 {
        x /=2
    }else {
        x = 3*x+1
    }
}
```

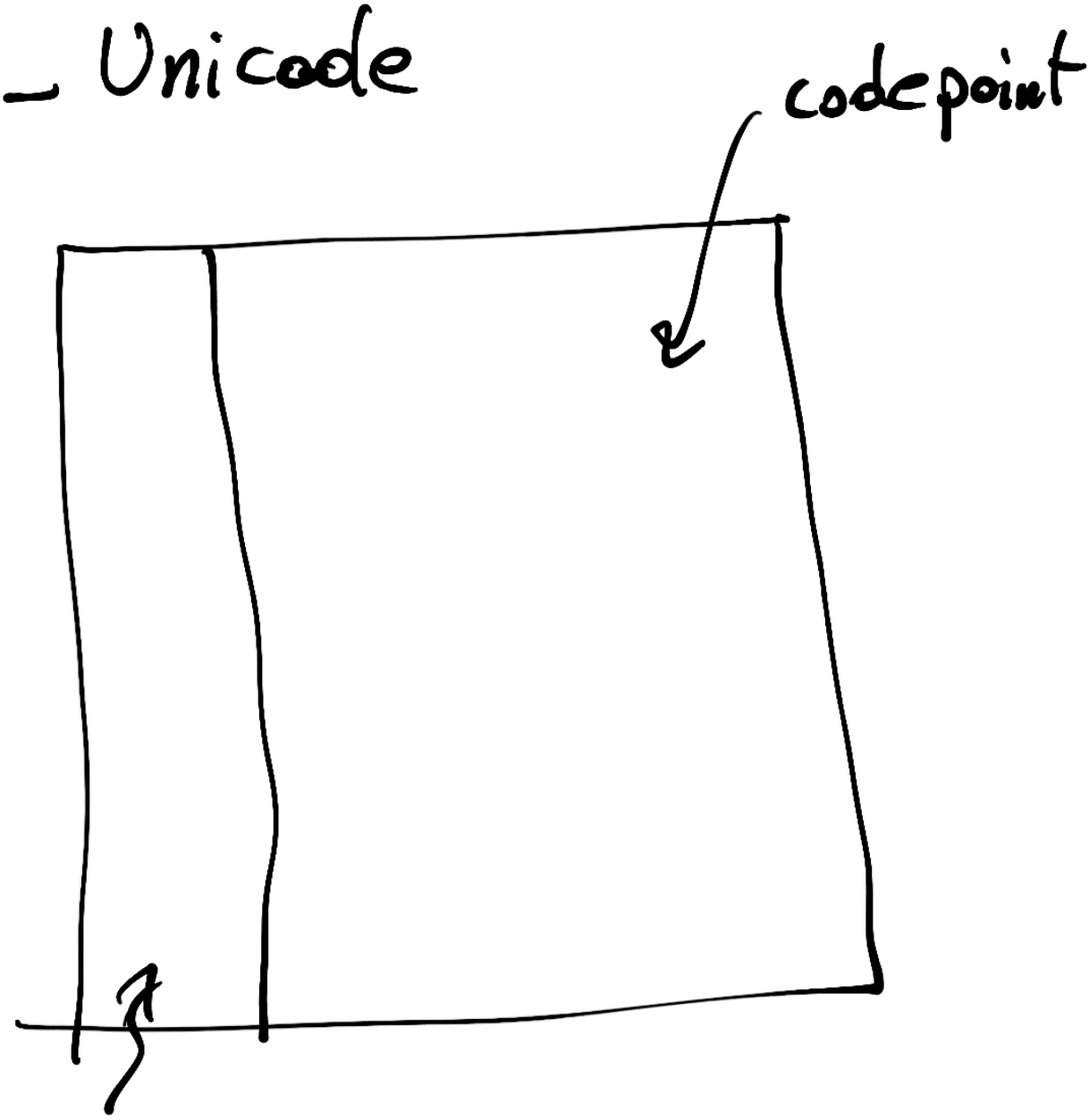
```
var x int
fmt.Scan(&x)
for x>1 {
    fmt.Println(x)

    if x%2 ==0 {
        x /=2
    }else {
        x = 3*x+1
    }
}
fmt.Println(1)
```

CARATTERI

- US-ASCII (7 bit)



_ Unicode



$17 \cdot 2^{16}$

plane

rune = int32

var x rune

```
x = 'A'
fmt.Println(x)                  | 65
→ fmt.Println(string(x))        | A
→ fmt.Println("%c", x)          | A
```

```
x = 'ラ'                        | 18210
x = '𠮷'                        | 8984
x = '\u2318'
```

↑ car. unicode con codice esad. 2318 (= dec. 8984)

x = '\U00002318'

'\t'	TAB
'\n'	NEW-LINE
'\r'	CARRIAGE RETURN
'\''	'
'\\'	\
'\"'	"

- ISO-8859-1 (o Latin1)
  - ↳ si usa un solo byte
    ma si rappr. solo
    i caratteri
    0 → 255

- UTF-16
  - ↳ usa 2 byte per
    carattere
    ma rappr. solo
    il piano 0 (BMP)

- UTF-8