

## Plancha 2: Punto flotante

2018 – Arquitectura de las Computadoras  
Licenciatura en Ciencias de la Computación

*Entrega: fecha a determinar*

1) Haga dos funciones o macros de C para extraer la fracción y el exponente de un `float` sin usar variables auxiliares.

Sugerencia: utilice corrimientos de bits y máscaras. Luego use los tipos definidos en la cabecera `ieee754.h` para corroborar.

2) El siguiente programa muestra algunas cualidades de *NaN* (*Not A Number*) y la función `isnan` de C, que indica si un flotante es *NaN*.

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float g = 0.0;
    float f = 0.0 / g;
    printf("f: %f\n", f);
    // ADVERTENCIA: 'NaN' es una extensión de GCC.
    if (f == NaN)
        printf("Es NaN\n");
    if (isnan(f))
        printf("isNaN dice que sí\n");
    return 0;
}
```

1. El programa muestra que comparar con `NaN` retorna siempre falso y para saber si una operación dio *NaN* se puede usar `isnan`. Utilizando las funciones del ejercicio anterior, implemente una función `myisnan` que haga lo mismo que la función `isnan` de C.
2. Implemente otra función, `myisnan2`, que haga lo mismo pero utilizando solo una comparación y sin operaciones de bits.
3. ¿Ocurre lo mismo con  $+\infty$ ? ¿Qué pasa si se suma un valor a  $+\infty$ ?

3) Convierta a `double` y `float` norma *IEEE 754* la constante número de Avogadro:  $N = 6,02252 \times 10^{23}$ . Realice el cálculo de manera explícita y luego corrobore el resultado mediante un algoritmo a partir del ejercicio anterior. Tenga en cuenta que  $\log_b x = n \Leftrightarrow x = b^n$ .

4) Sabiendo que un número en punto flotante puede representarse usando base  $b$  y exceso  $q$  como:

$$(signo, f, e) = signo \times f \times b^{e-q} \quad |f| < 1,$$

implemente la suma y producto de números base 2 y exceso 30000. Use 18 bits para  $f$  y 16 para  $e$ . ¿Cuáles números son el mayor y el menor en esta representación?

Sugerencia: use campos de bits.

**5)** Realice el procedimiento de suma (simple precisión) del número  $1,75 \times 2^{-79}$  con el siguiente número expresado en IEEE 754:

0|00110000|101000000000000000000000