INFORMATICA I

Representación de los número reales IEEE (754)

Ing.Juan Carlos Cuttitta

Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Departamento de Ingeniería Electrónica

7 de mayo de 2019

Repaso de notación científica !!!!

$$\mathbb{R} = c \cdot b^e$$

- $\mathbb{R}$ representa a todos los números comprendidos en el intervalo (-∞ a +∞).
- c es el coeficiente y está formado por un número real con un solo digito entero seguido de una coma y varios digitos fraccionarios.
- b es la base del sistema de numeración
  - 16 en el hexadecimal
  - 10 en el decimal
  - 8 en el octal
  - 2 en el binario
- e es el exponente entero el cual eleva la base a una potencia.

Repaso de notación científica !!!!

El coeficiente tiene una cantidad determinada de digitos significativos y ésta cantidad indica la precisión del número representado. Cuanto más dígito, mayor será la precisión.

Ejemplo con el número de Euler:

$$\underbrace{2{,}71}_{c} \cdot \underbrace{10}_{b}^{\overbrace{0}^{e}} \longrightarrow \text{menor precisión}$$

$$2{,}718281 \cdot 10^{0} \longrightarrow \text{mayor precisión}$$

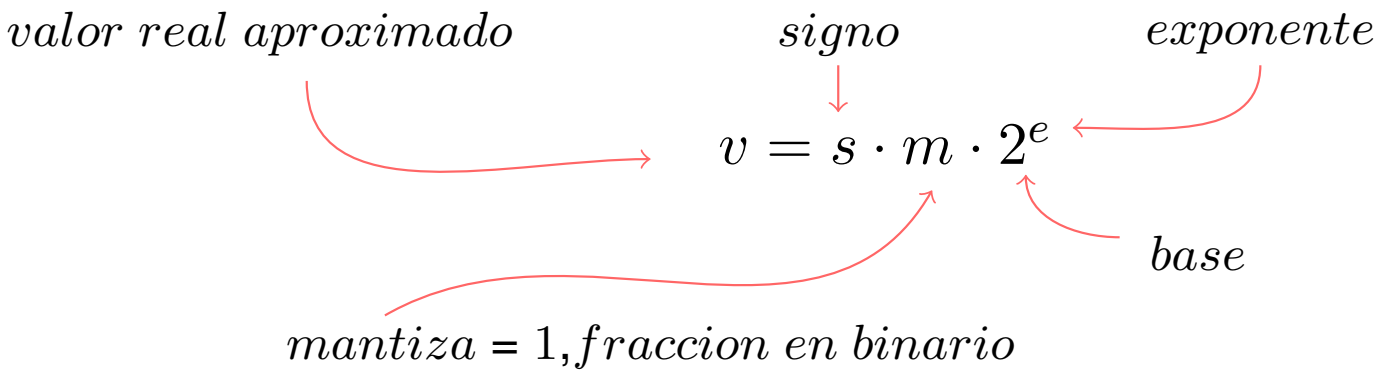
Repaso de notación científica !!!!

Al multiplicar el coeficiente (*c*) por la base (*b*) elevada a una potencia entera (*e*), lo que estamos haciendo es desplazando la coma del coeficiente (*c*) tantas posiciones como indique el exponente (e). La coma se desplaza hacia la *derecha* si el exponente es *positivo* y hacia la *izquierda* si es *negativo*.
Ejemplos:

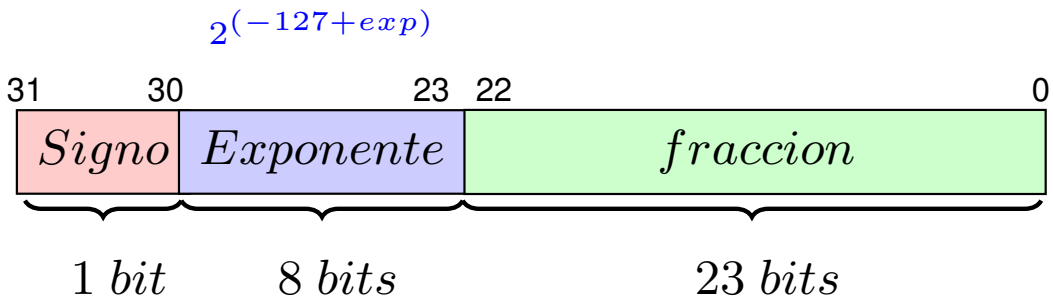
Notación científica	**Representación del n° $\mathbb{R}$**
$3{,}14159 \cdot 10^{-2}$	0.0314159
$3{,}14159 \cdot 10^{-1}$	0.314159
$3{,}14159 \cdot 10^{0}$	3.14159
$3{,}14159 \cdot 10^{1}$	31.4159
$3{,}14159 \cdot 10^{2}$	314.159

estándar del *IEEE* para coma flotante *(IEEE 754)*

- El coma flotante permite representar una cantidad limitada de digitos de un $\mathbb{R}$
- Un número $\mathbb{R}$ no se podrá representar con total precisión sino como una aproximación.
- La precisión dependerá de la cantidad de digitos significativos que tenga la representación en coma flotante.



Como $1 \leq m < 2$ y a fin de normalizar la representación binaria, en la mantiza sólo se coloca la fracción binaria y se asume implicitamente al 1 que está ubicado a la izquierda de la coma.
Como se puede observar en la figura, necesitamos obtener el signo, la fracción y el exponente; donde este último se representa en binario desplazado.



Ejemplo práctico con el número -37.625
Convertimos la parte entera a binario como se ve en la siguiente tabla (en nuestro ejemplo es 37).

división	resultado	resto
$37 \div 2$	18	1
$18 \div 2$	9	0
$9 \div 2$	4	1
$4 \div 2$	2	0
$2 \div 2$	1	0
$1 \div 2$	0	1

$$(37)_{10} = (100101)_2$$

Luego convertimos la parte decimal a binario como se ve en la siguiente tabla (en nuestro ejemplo es 0.625).

multiplicación	**resultado**	**parte entera**
$0{,}625 * 2$	1.25	1
$0{,}25 * 2$	0.5	0
$0{,}5 * 2$	1	1

De esta forma queda determinada la representacion del número en formato binario.

$$(-37{,}625)_{10} = (-100101{,}101)_2$$

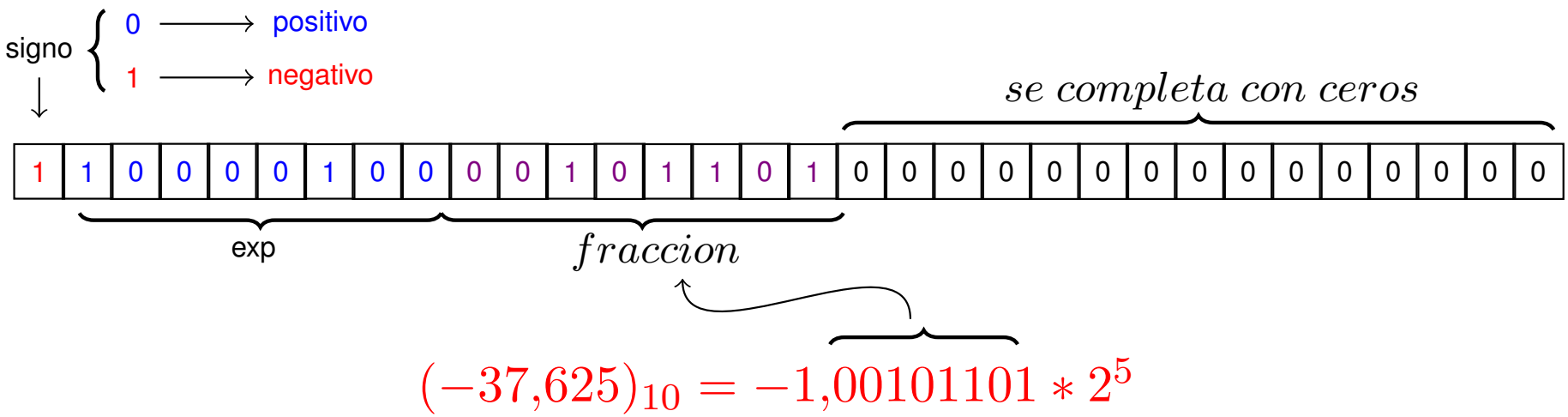
El resultado se representa en notación científica.

$$-1{,}00101101 * 2^5$$

Para obtener el valor del exponente se despeja exp de la siguiente igualdad.

$$2^{(-127+exp)} = 2^5$$
$$-127 + exp = 5$$
$$exp = 5 + 127$$
$$exp = 132$$
$$(132)_{10} = (10000100)_2$$

estándar del *IEEE* para coma flotante *(IEEE 754)*

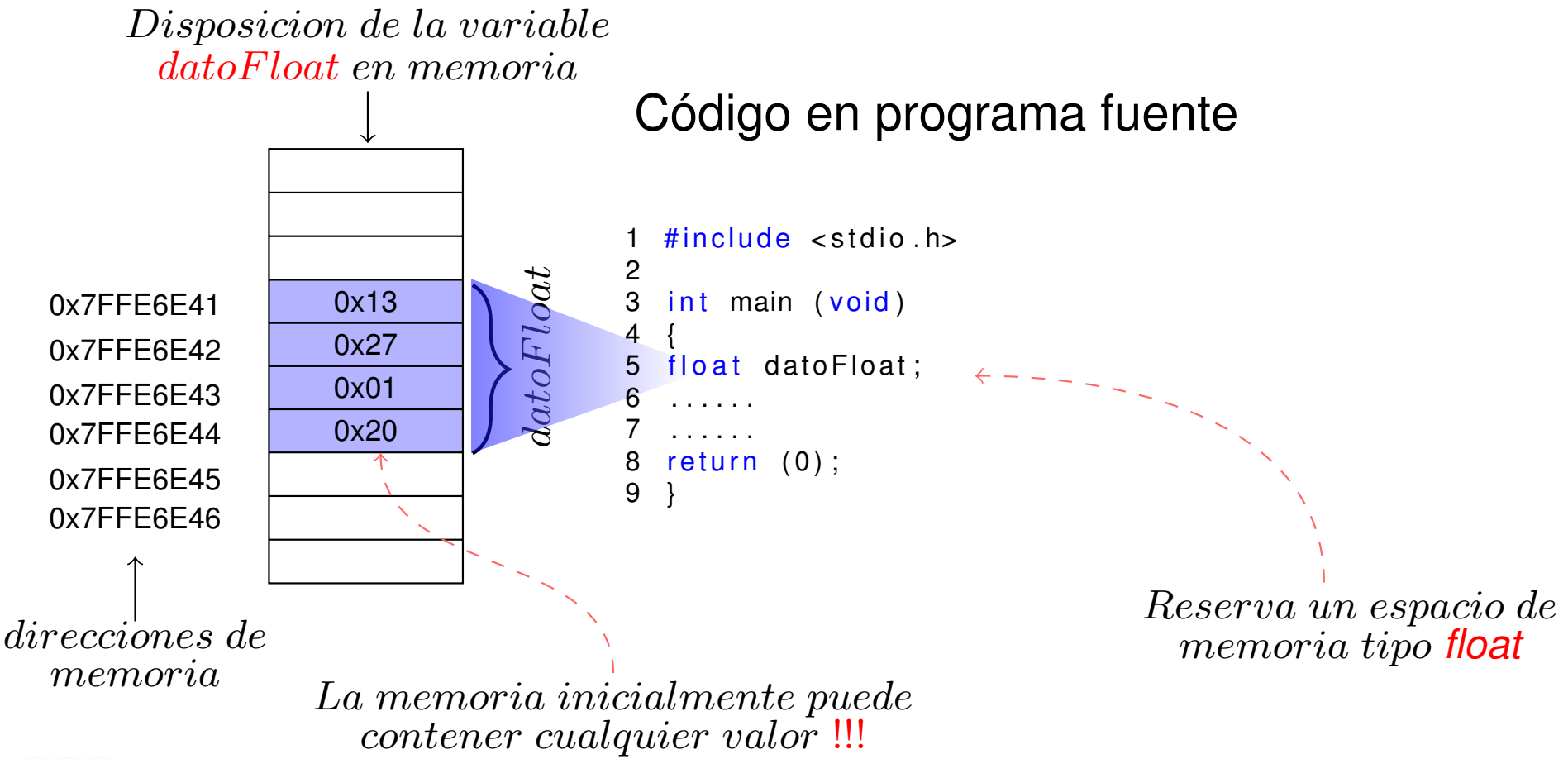


El rango de precisión aproximado para 32 bits normalizados es

binario	**decimal**
2^{-126} a 2^{127}	$1{,}18 \cdot 10^{-38}$ a $3{,}4 \cdot 10^{38}$

Función scanf para tipo de dato *float*

Arquitectura X86 32 bits

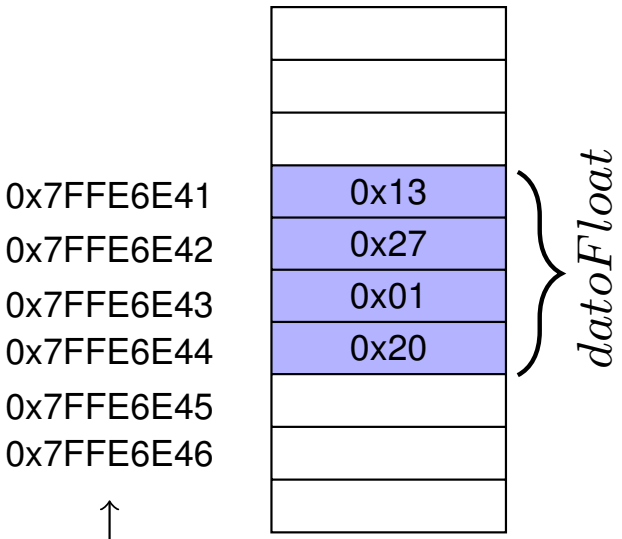


Código en programa fuente

```
1 #include <stdio.h>
2
3 int main (void)
4 {
5 float datoFloat;
6 ......
7 ......
8 return (0);
9 }
```

Función scanf para tipo de dato *float*

Disposicion de la variable datoFloat en memoria

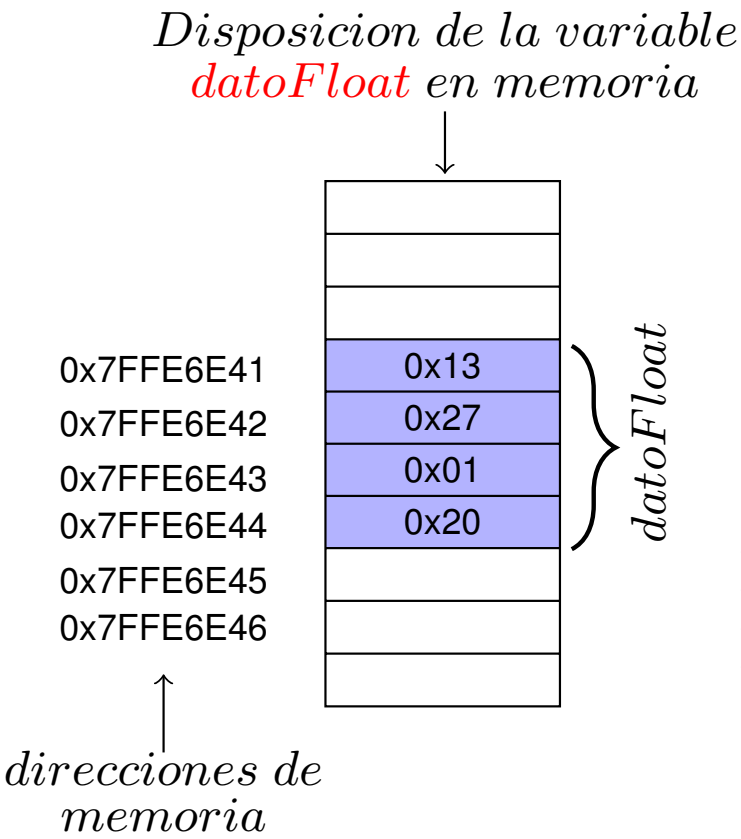


Código en programa fuente

```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  float datoFloat;
6
7  //— Ingreso de datos —
8    printf("Ingrese valor : ");
9    ......
10   ......
11 return (0);
12 }
```

Imprime en pantalla una leyenda para el usuario

Función scanf para tipo de dato ***float***

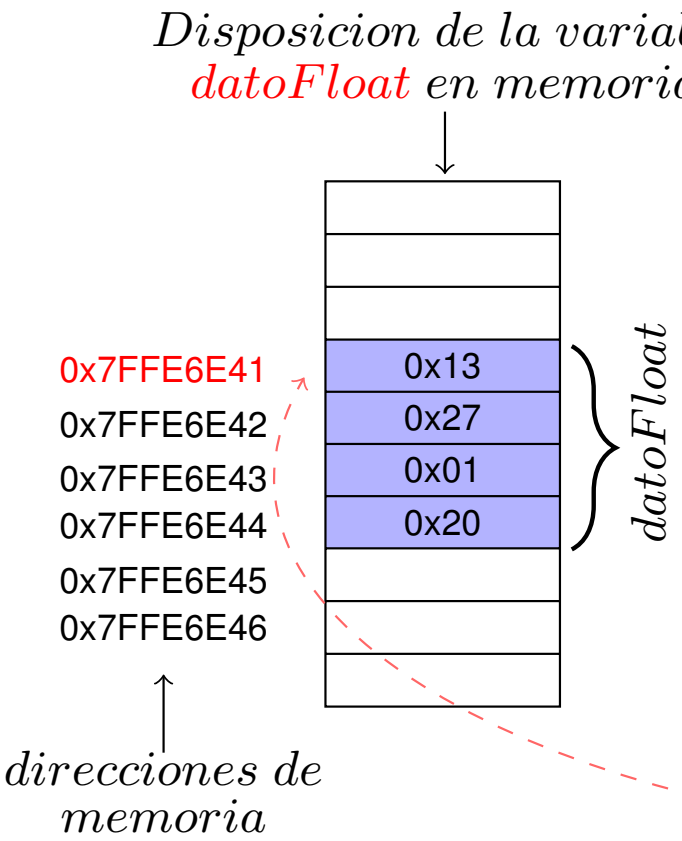


Código en programa fuente

```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  float datoFloat;
6
7  //— Ingreso de datos —
8    printf("Ingrese valor : ");
9    scanf("%f", &datoFloat);
10   ......
11   ......
12 return (0);
13 }
```

Indica el tipo de dato que toma del teclado

Función scanf para tipo de dato *float*

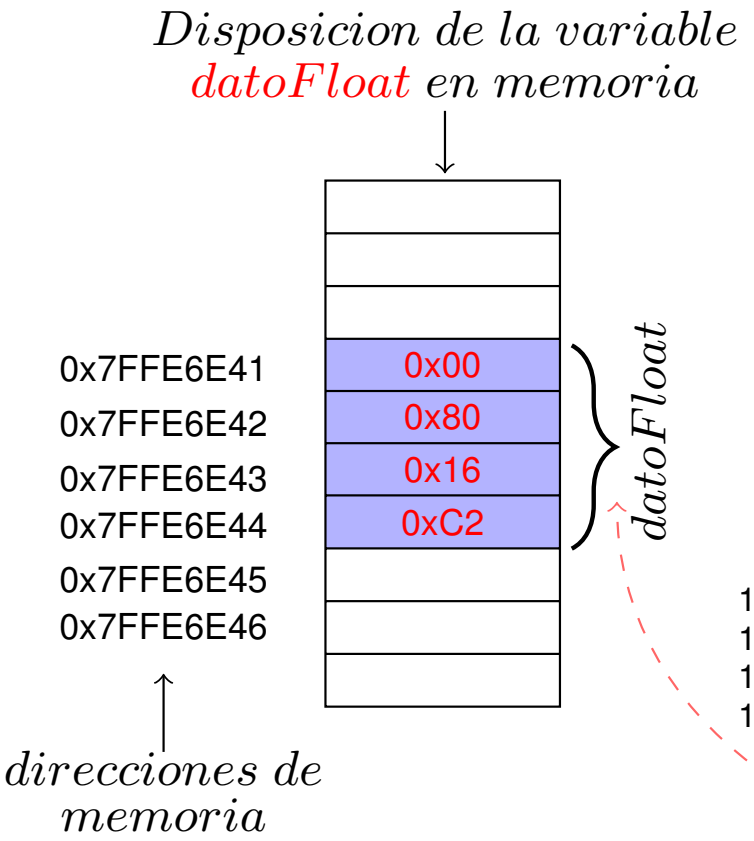


Código en programa fuente

```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  float datoFloat;
6
7  //— Ingreso de datos —
8    printf("Ingrese valor : ");
9    scanf("%f", &datoFloat);
10   ......
11   ......
12 return (0);
13 }
```

Indica la direccion de memoria donde va a guardar el dato

Función scanf para tipo de dato ***float***



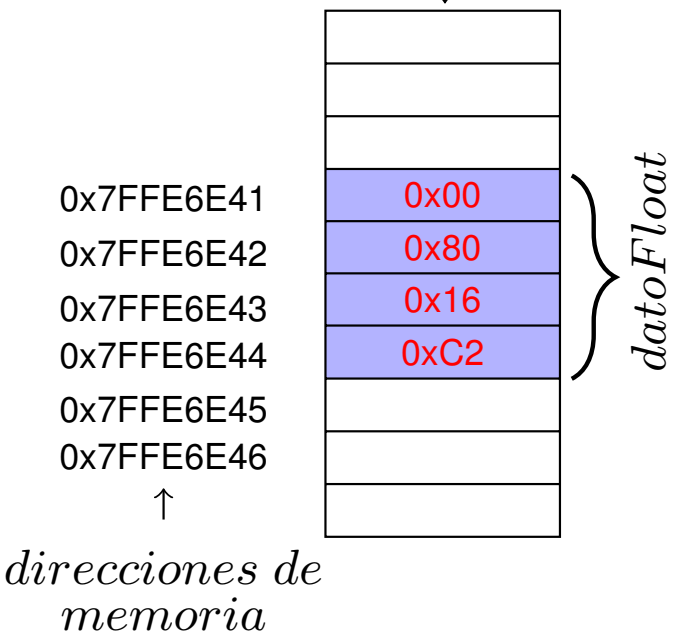
Código en programa fuente

```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  float datoFloat;
6
7  //— Ingreso de datos —
8    printf("Ingrese valor : ");
9    scanf("%f", &datoFloat);
10   ......
11   ......
12 return (0);
13 }
```

El dato ingresado fue el numero $-37{,}625$

Función printf para tipo de dato *float*

Disposicion de la variable datoFloat en memoria

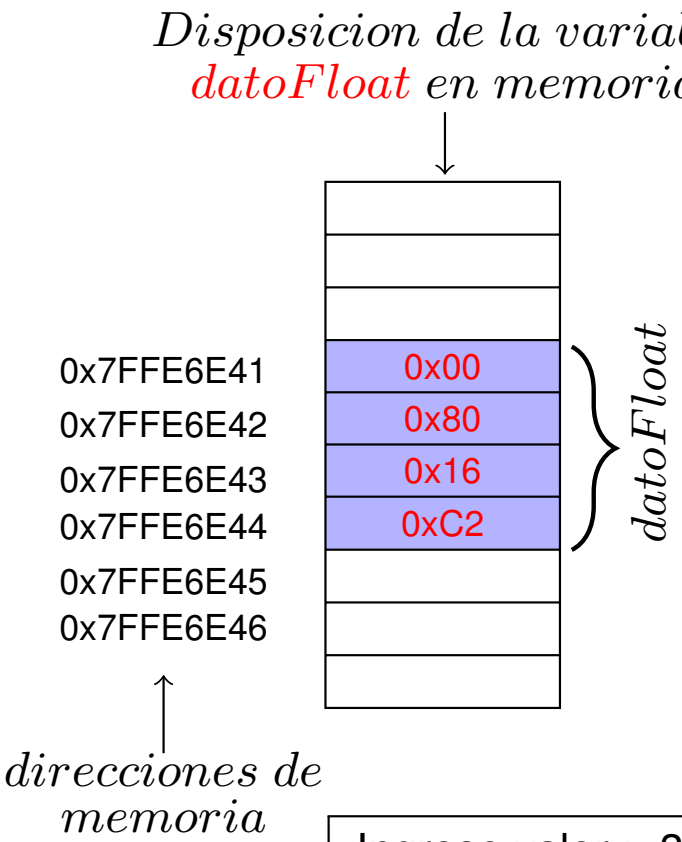


Código en programa fuente

```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  float datoFloat;
6
7  //-- Ingreso de datos --
8    printf("Ingrese valor : ");
9    scanf("%f", &datoFloat);
10 //-- Impresión de datos --
11   printf("size del float:\t%d\r\n",(int)sizeof(datoFloat));
12   ......
13 return (0);
14 }
```

```
Ingrese valor : -37.625
size del float          :   4
```

Función printf para tipo de dato *float*

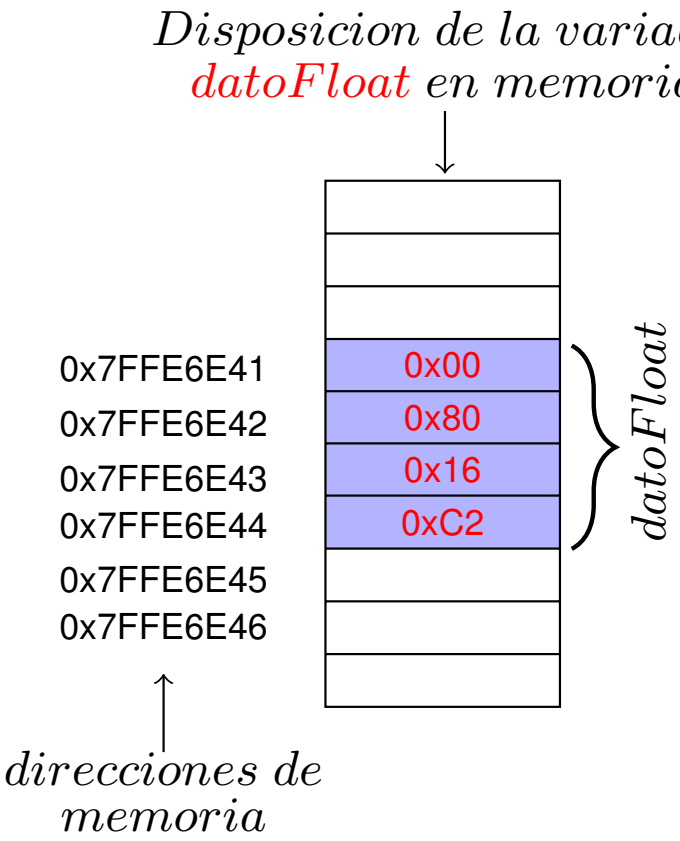


Código en programa fuente

```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  float datoFloat;
6
7  //—— Ingreso de datos ——
8    printf("Ingrese valor : ");
9    scanf("%f", &datoFloat);
10 //—— Impresión de datos ——
11   printf("size del float:\t%d\r\n",(int)sizeof(datoFloat));
12   printf("el dato en real:\t%f\r\n",datoFloat);
13   ......
14 return (0);
15 }
```

```
Ingrese valor : -37.625
size del float      :   4
el dato en real     :   -37.625000
```

Función printf para tipo de dato *float*

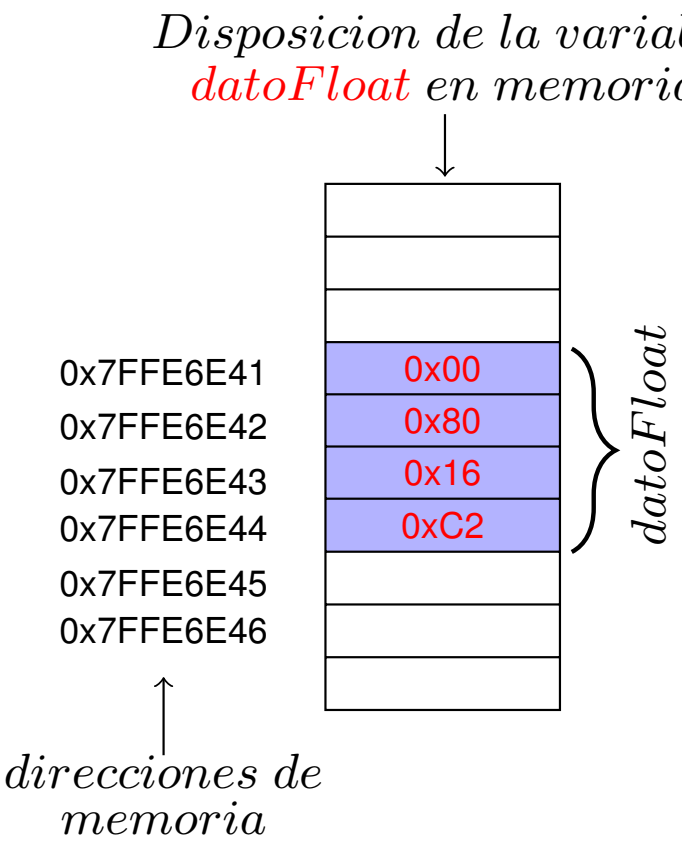


Código en programa fuente

```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  float datoFloat;
6
7  //—— Ingreso de datos ——
8    printf("Ingrese valor : ");
9    scanf("%f", &datoFloat);
10 //—— Impresión de datos ——
11   printf("size del float:\t%d\r\n",(int)sizeof(datoFloat));
12   printf("el dato en real:\t%f\r\n",datoFloat);
13   printf("el dato en real:\t%0.2f\r\n",datoFloat);
14   ......
15 return (0);
16 }
```

```
Ingrese valor : -37.625
size del float          :   4
el dato en real         :   -37.625000
el dato en real         :   -37.62
```

Función printf para tipo de dato *float*



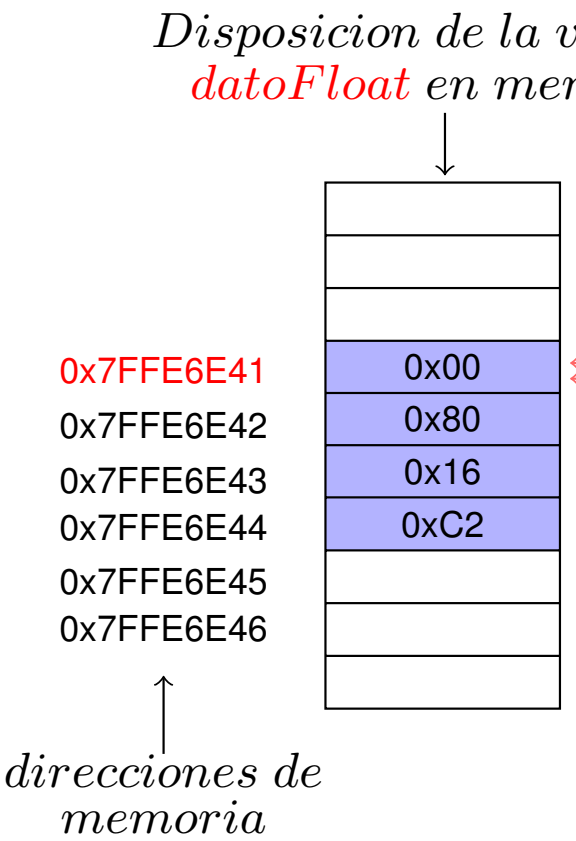
Código en programa fuente

```c
#include <stdio.h>

int main (void)
{
float datoFloat;
//— Ingreso de datos —
  printf("Ingrese valor : ");
  scanf("%f", &datoFloat);
//— Impresión de datos —
  printf("size del float:\t%d\r\n",(int)sizeof(datoFloat));
  printf("el dato en real:\t%f\r\n",datoFloat);
  printf("el dato en real:\t%0.2f\r\n",datoFloat);
  printf("dirección en memoria:\t%p\r\n", &datoFloat);
return (0);
}
```

```
Ingrese valor : -37.625
size del float          :   4
el dato en real         :   -37.625000
el dato en real         :   -37.62
dirección en memoria    :   0x7FFE6E41
```

Verificación de datos tipo *float* guardados en memoria




```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  unsigned char *p;
6  float datoFloat;
7   p=(unsigned char*)&datoFloat;
8  //-- Ingreso de datos --
9    printf("Ingrese valor : ");
10   scanf("%f", &datoFloat);
11 //-- Impresión de datos --
12   printf("el dato en float       :\t%f\r\n",datoFloat);
13   printf("dirección en memoria   :\t%p\r\n",&datoFloat);
14   printf("contenido de p es      :\t%p\r\n",&datoFloat);
15   printf("el dato en *p es       :\t%#x\r\n",*p);
16   printf("el dato en *(p+1)      :\t%#x\r\n",*(p+1));
17   printf("el dato en *(p+2)      :\t%#x\r\n",*(p+2));
18   printf("el dato en *(p+3)      :\t%#x\r\n",*(p+3));
19 return (0);
20 }
```

```
Ingrese valor : -37.625
el dato en float        :   -37.625000
dirección en memoria    :   0x7FFE6E41
contenido de p es       :   0x7FFE6E41
el dato en *p es        :   0x00
el dato en *(p+1) es    :   0x80
el dato en *(p+2) es    :   0x16
el dato en *(p+3) es    :   0xC2
```

Verificación de datos tipo *float* guardados en memoria

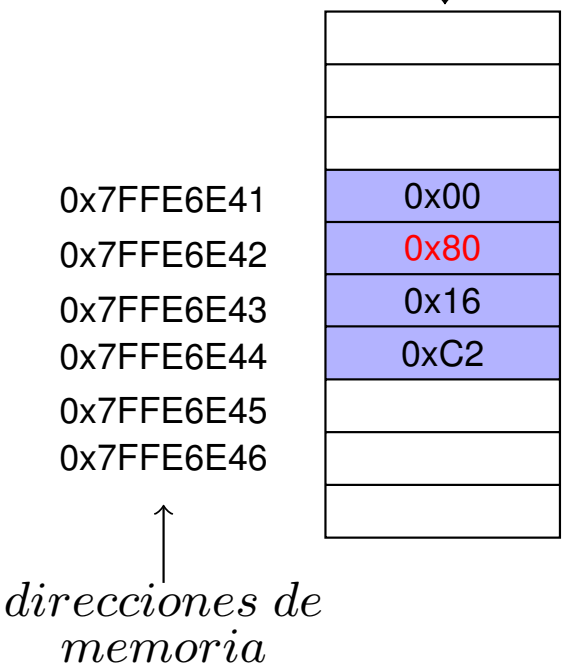
Disposicion de la variable datoFloat en memoria


```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  unsigned char *p;
6  float datoFloat;
7    p=(unsigned char*)&datoFloat;
8  //-- Ingreso de datos --
9    printf("Ingrese valor : ");
10   scanf("%f", &datoFloat);
11 //-- Impresión de datos --
12   printf("el dato en float       :\t%f\r\n",datoFloat);
13   printf("dirección en memoria   :\t%p\r\n",&datoFloat);
14   printf("contenido de p es      :\t%p\r\n",&datoFloat);
15   printf("el dato en *p es       :\t%#x\r\n",*p);
16   printf("el dato en *(p+1)      :\t%#x\r\n",*(p+1));
17   printf("el dato en *(p+2)      :\t%#x\r\n",*(p+2));
18   printf("el dato en *(p+3)      :\t%#x\r\n",*(p+3));
19 return (0);
20 }
```

```
Ingrese valor : -37.625
el dato en float        :   -37.625000
dirección en memoria    :   0x7FFE6E41
contenido de p es       :   0x7FFE6E41
el dato en *p es        :   0x00
el dato en *(p+1) es    :   0x80
el dato en *(p+2) es    :   0x16
el dato en *(p+3) es    :   0xC2
```

Verificación de datos tipo *float* guardados en memoria

Disposicion de la variable datoFloat en memoria




```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  unsigned char *p;
6  float datoFloat;
7    p=(unsigned char*)&datoFloat;
8  //-- Ingreso de datos --
9    printf("Ingrese valor : ");
10   scanf("%f", &datoFloat);
11 //-- Impresión de datos --
12   printf("el dato en float       :\t%f\r\n",datoFloat);
13   printf("dirección en memoria   :\t%p\r\n",&datoFloat);
14   printf("contenido de p es      :\t%p\r\n",&datoFloat);
15   printf("el dato en *p es       :\t%#x\r\n",*p);
16   printf("el dato en *(p+1)      :\t%#x\r\n",*(p+1));
17   printf("el dato en *(p+2)      :\t%#x\r\n",*(p+2));
18   printf("el dato en *(p+3)      :\t%#x\r\n",*(p+3));
19 return (0);
20 }
```

```
Ingrese valor : -37.625
el dato en float        :   -37.625000
dirección en memoria    :   0x7FFE6E41
contenido de p es       :   0x7FFE6E41
el dato en *p es        :   0x00
el dato en *(p+1) es    :   0x80
el dato en *(p+2) es    :   0x16
el dato en *(p+3) es    :   0xC2
```

Verificación de datos tipo *float* guardados en memoria

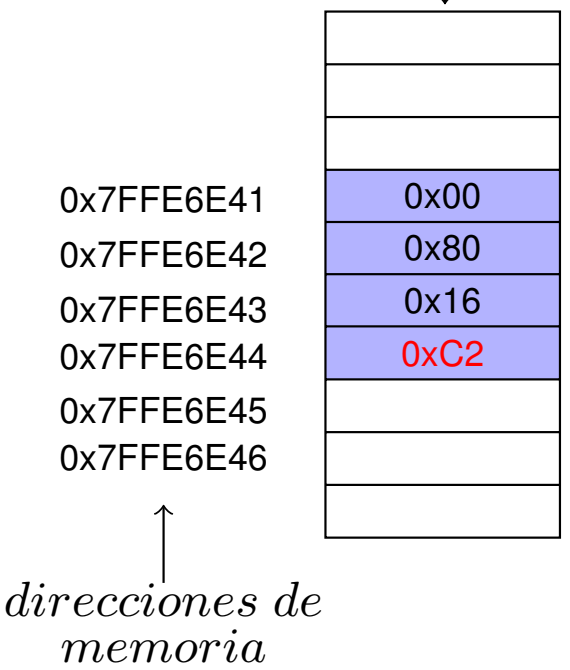
Disposicion de la variable datoFloat en memoria


```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  unsigned char *p;
6  float datoFloat;
7    p=(unsigned char*)&datoFloat;
8  //-- Ingreso de datos --
9    printf("Ingrese valor : ");
10   scanf("%f", &datoFloat);
11 //-- Impresión de datos --
12   printf("el dato en float       :\t%f\r\n",datoFloat);
13   printf("dirección en memoria   :\t%p\r\n",&datoFloat);
14   printf("contenido de p es      :\t%p\r\n",&datoFloat);
15   printf("el dato en *p es       :\t%#x\r\n",*p);
16   printf("el dato en *(p+1)      :\t%#x\r\n",*(p+1));
17   printf("el dato en *(p+2)      :\t%#x\r\n",*(p+2));
18   printf("el dato en *(p+3)      :\t%#x\r\n",*(p+3));
19 return (0);
20 }
```

```
Ingrese valor : -37.625
el dato en float        :   -37.625000
dirección en memoria    :   0x7FFE6E41
contenido de p es       :   0x7FFE6E41
el dato en *p es        :   0x00
el dato en *(p+1) es    :   0x80
el dato en *(p+2) es    :   0x16
el dato en *(p+3) es    :   0xC2
```

Verificación de datos tipo *float* guardados en memoria

Disposicion de la variable datoFloat en memoria




```
1  #include <stdio.h>
2
3  int main (void)
4  {
5  unsigned char *p;
6  float datoFloat;
7    p=(unsigned char*)&datoFloat;
8  //-- Ingreso de datos --
9    printf("Ingrese valor : ");
10   scanf("%f", &datoFloat);
11 //-- Impresión de datos --
12   printf("el dato en float       :\t%f\r\n",datoFloat);
13   printf("dirección en memoria   :\t%p\r\n",&datoFloat);
14   printf("contenido de p es      :\t%p\r\n",&datoFloat);
15   printf("el dato en *p es       :\t%#x\r\n",*p);
16   printf("el dato en *(p+1)      :\t%#x\r\n",*(p+1));
17   printf("el dato en *(p+2)      :\t%#x\r\n",*(p+2));
18   printf("el dato en *(p+3)      :\t%#x\r\n",*(p+3));
19 return (0);
20 }
```

```
Ingrese valor : -37.625
el dato en float        :   -37.625000
dirección en memoria    :   0x7FFE6E41
contenido de p es       :   0x7FFE6E41
el dato en *p es        :   0x00
el dato en *(p+1) es    :   0x80
el dato en *(p+2) es    :   0x16
el dato en *(p+3) es    :   0xC2
```