Estructura de computadores

Practica 1

Javier Bautista Rosell NIA:100315139
correo: 100315139@alumnos.uc3m.es

Alejandro Blanco Samaniego NIA: 100315850
correo: 100315850@alumnos.uc3m.es

Índice

Ejercicio 1, E1-Series

a) como nos dice el programa al ejecutarlo con precisión simple el numero máximo de iteraciones son 26
b) Con precisión doble el numero máximo de iteraciones son 55
c) El tamaño de int son 32 bits
d) El tamaño del long son 64 bits. En nuestro caso hemos utilizado una arquitectura de 64 bits
e) Si utilizáramos una arquitectura de 32 bits el long no tendría 64 bits si no 32, porque la arquitectura de 32 bits no alcanza los 64 bits, por ende si utilizamos una arquitectura de 64 bits el long tendrá 64 bits.

He utilizado un while para calcular el valor de la serie y cuando un valor fuera siguiente al anterior se parara para saber el numero de iteraciones que habíamos hecho.
Para las longitudes de int y long hice un while multiplicando por 2 comenzando desde el 1 con un contador hasta que dio un numero negativo.

El resultado del programa es el siguiente:

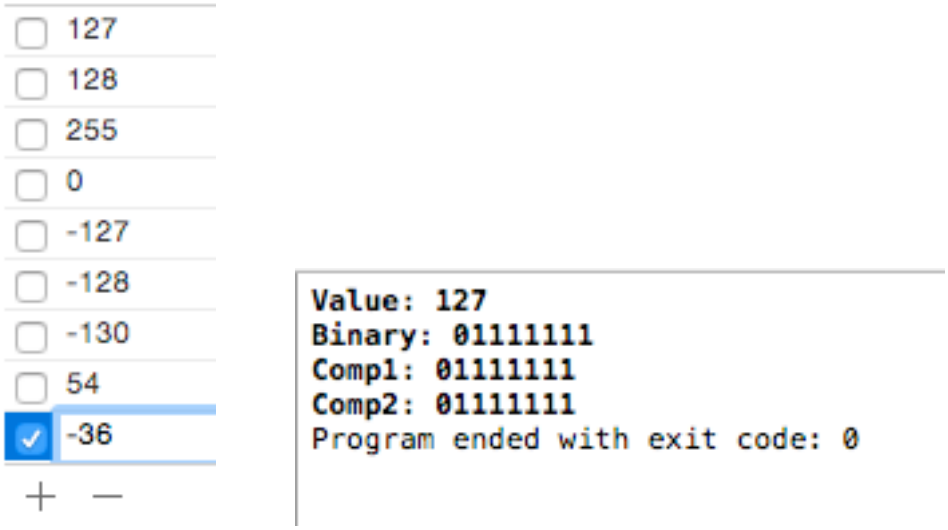
```
Executing IEEE 754 series..

el valor de la serie no varia a partir del Sn donde n= 55 si la
variable es double
si la variable fuera float n seria=26

Executing integer series..
la variable int tiene 32 bits
la variable long tiene 64 bits
Program ended with exit code: 0
```

Ejercicio 2

En este ejercicio se ha usado el mismo esquema para las tres funciones, ya que no nos deja el enunciado devolver un valor en las funciones. Por ello, repetimos el código con sus correspondientes comentarios para cada función.
En la primera, convertimos un número decimal a binario, con 8 bits, lo cual quiere decir que solo podremos representar de 0 a 255, así que al comienzo excluyo el resto de valores, comentando que no es representable.
Para convertir los representables, comienzo rellenando el vector con unos y ceros correspondientes por la derecha, con el primer bucle while. Esto lo conseguimos con el método de dividir entre 2.
Luego, me encargo de poner ceros a la izquierda con el siguiente bucle for. De esta manera, resolvemos la primera función.
Para la función que nos pide pasar de decimal a complemento a1, realizamos la misma operación que en la anterior función (teniendo cuidado de excluir los valores que no estén entre -127 y 127) y tras ello, para los valores negativos, trabajamos como si fueran positivos y sustituimos luego los unos por 0.
Para la función que nos pide pasar de decimal a complemento a2, realizamos la misma operación que en la anterior función (teniendo cuidado de excluir esta vez los valores que no estén entre -128 y 127) y tras ello, para los valores negativos, sumamos un bit lógico (paso de Ca1 a Ca2). El programa compila y da resultado en todos los valores sometidos a prueba.
La siguiente imagen es el orden de los valores que hemos probado.



```
Not a formatted number.
Program ended with exit code: 255
```

```
Not a formatted number.
Program ended with exit code: 255
```

```
Value: 0
Binary: 00000000
Comp1: 00000000
Comp2: 00000000
Program ended with exit code: 0
```

```
Value: -127
Binary: Este numero no es representable en binario natural con 8 bits00000000
Comp1: 10000000
Comp2: 10000001
Program ended with exit code: 0
```

```
Value: -128
Binary: Este numero no es representable en binario natural con 8 bits00000000
Comp1: Este numero no es representable en complemento A1 con 8 bits00000000
Comp2: 10000000
Program ended with exit code: 0
```

```
Not a formatted number.
Program ended with exit code: 255
```

```
Value: 54
Binary: 00110110
Comp1: 00110110
Comp2: 00110110
Program ended with exit code: 0
```

```
Value: -36
Binary: Este numero no es representable en binario natural con 8 bits00000000
Comp1: 11011011
Comp2: 11011100
Program ended with exit code: 0
```

Ejercicio3 E3-FloatToIEEE754

En este ejercicio he aplicado mis conocimientos de IEEE 754 para conseguir pasar un numero de coma flotante a IEEE754 de 32 bits teniendo en cuenta tanto el signo del numero, como el exponente y el exceso de 127 que hay que ponerle al pasarlo a IEE754 de 32 bits.
He rellenado el vector dado según me ha parecido más fácil y correcto y no se si será lo más simple posible pero a mi parecer si lo es.
Para comprobar que funciona he hecho algunas pruebas:

```
Value: 25.4500007629394531

 exponente = 4
0 10000011 10010111001100110011010
Program ended with exit code: 0
```

```
Value: 0.5000000000000000

 exponente = -1
0 01111110 00000000000000000000000
Program ended with exit code: 0
```

```
Value: 425.0000000000000000

 exponente = 8
0 10000111 10101001000000000000000
Program ended with exit code: 0
```

```
Value: 2.0000000000000000

 exponente = 1
0 10000000 00000000000000000000000
Program ended with exit code: 0
```

```
Value: -425.0000000000000000

 exponente = 8
1 10000111 10101001000000000000000
Program ended with exit code: 0
```

Ejercicio 4 IEEE 754 To Float:

En este ejercicio he aplicado los mismos conocimientos que en el ejercicio anterior para hacer lo contrario. No he conseguido poner el valor que me sale en vuestro vector Value, espero que eso tampoco sea un gran fallo dado que por pantalla muestro el valor que da.

Algunas pruebas realizadas en el ejercicio:

```
1 01111110 00000000000000000000000
exponente = -1
el valor es = -0.500000
```

```
0 01111110 00000000000000000000000
exponente = -1
el valor es = 0.500000
```

```
1 11111111 11111111111111111111111
exponente = 128
 -1.000000
el valor es = nan
```

Ejercicio 5

En este ejercicio, no comprendimos muy bien el enunciado, ya que en este no explica en que formato se devuelve el resultado. Entendemos que en IEE 754. Además, nos agregan una función que no explica el enunciado, la cual tampoco entendemos por qué está ahí. Se trata de una función que correspondería al ejercicio 4. Igualmente, la hemos rellenado como creemos que corresponde.
Lo planteamos de la siguiente manera. Recibimos el vector en IEE 754, y si es positivo, le restamos un bit lógico, y si fuese negativo, se lo sumamos. Esta suma afectaría a los bits de la mantisa y exponente, en caso de que la mantisa fuese todo 0 en caso de positivos o todo 1 en caso de negativos. Nunca afectaría al signo.
También, antes, nos aseguramos de conocer y excluir los casos especiales; es decir, si es exponente 0 y mantisa 0, pasamos de 0 a -0, para comenzar a trabajar con negativo, o de -0 a -0, que nos daría lo mismo. Si es exponente 0 y mantisa no 0, se señala que es un número no normalizado. Si es exponente 255 y mantisa 0, se señala que es un caso NAN (Not a Number). Y si es exponente 255 y mantisa 1, se fija en el signo para señalar +infinito o -infinito.

Ejercicio 6

En este ejercicio también consideramos incorrecto el enunciado, ya que dice que lo que nos dan es un estándar IEE 754 y en realidad lo que nos da el ejercicio y la función es un float decimal, con el cual trabajamos para su conversión directa al IEE 754 de 64 bits (precisión doble).
Su método de resolución idéntico al del ejercicio 3, viéndose modificados solo los valores numéricos que caracterizan la representación en IEE 754. Es decir, donde antes teníamos un exponente de 8 bits y mantisa de 23, ahora tenemos un exponente de 11 bits y mantisa de 52. Lo que hemos hecho entonces ha sido sustituir los 8 por 11; 9 por 12; 127 por 1023; y 31 por 63. El programa compila y da resultado en todos los valores sometidos a prueba.
También, antes, al igual que en el ejercicio 5, nos aseguramos de conocer y excluir los casos especiales; si es exponente 0 y mantisa no 1, se señala que es un número no normalizado. Si es exponente 255 y mantisa 0, se señala que es un caso NAN (Not a Number). Y si es exponente 255 y mantisa 1, se fija en el signo para señalar +infinito o -infinito.

```
Value: 14725.000000000000000

 exponente = 13
0 10000001100 11001100001010000000000000000000000000000000000000000000
Program ended with exit code: 0
```

```
Value: 0.250000000000000

 exponente = -2
0 01111111101 00000000000000000000000000000000000000000000000000000000
Program ended with exit code: 0
```

```
Value: -14725.000000000000000

 exponente = 13
1 10000001100 11001100001010000000000000000000000000000000000000000000
Program ended with exit code: 0
```

```
Value: 10000000000.000000000000000

 exponente = 0
0 01111111111 11111111111111111111111111111111111111111111111111111111
Program ended with exit code: 0
```