Práctica. Introducción a la programación en C y lenguaje ensamblador

1. Objetivos de la práctica

El objetivo de la práctica consiste en entender el formato de representación de los números en coma flotante en el computador. Para ello se propone el desarrollo de una serie de programas en lenguaje C y ensamblador del MIPS32.

Para el desarrollo de la práctica es necesario que el alumno repase:

- Los principales aspectos del lenguaje de programación C:
- Los diferentes operadores que ofrece C para trabajar con bits. En el siguiente enlace se puede obtener información sobre los diferentes operadores de bit de C.

  http://en.wikipedia.org/wiki/Bitwise_operations_in_C
- La representación de números en coma flotante según el estándar IEEE754 de precisión simple.
- Los principales aspectos del ensamblador de MIPS 32 y el simulador QTSPIM utilizado para el desarrollo de la parte de ensamblador.

Ejercicio 1

Escriba en lenguaje C un programa lea dos matrices cuadradas A y B de números de tipo *float*, y genere a partir de ellas otras dos matrices cuadradas (C y D) de la misma dimensión cuyo contenido se describirá a continuación. El programa, denominado sumarMin, se invocará de la siguiente manera desde la línea de mandatos:

```
sumarMin  N  ficheroA  ficheroB  ficheroC  ficheroD
```

Donde `N` es un número que representa la dimensión de las cuatro matrices y `ficheroA`, `ficheroB`, `ficheroC` y `ficheroD` son ficheros de texto que almacenan cada matriz y tienen el siguiente formato:

```
dato1  dato2  dato3 …….  datoNxN
```

Donde `datoi` almacena un número de tipo *float*. Los valores que puede tener cada dato pueden ser los siguientes:

- 0, 0.0, -0, -0.0 para representar el valor 0.
- NaN, para representar el valor Not a Number.
- +Inf, para representar + infinito.
- -Inf, para representar –infinito.
- Cualquier número de tipo *float* representable (ejemplos 2.3, 2.34e23, etc.). Se asumirá que los números almacenados son representables en una variable de tipo *float*.

Las matrices se almacenan en los ficheros por filas, un dato seguido de otro. En el material de apoyo se proporciona un ejemplo de matriz.

El objetivo es generar dos matrices C y D, de forma que C[i][j] se obtiene de la siguiente manera:

1. Si A[i][j] == ±0 y B[i][j]== ±0, entonces C[i][j] = +0
2. Si A[i][j] == NaN o B[i][j]== NaN, entonces C[i][j] = NaN
3. Si A[i][j] == ±Inf o B[i][j]== ±Inf, entonces C[i][j] = NaN
4. Si A[i][j] almacena un valor no normalizado (se considera que el valor 0 es normalizado) y B[i][j] almacena también un valor no normalizado, entonces C[i][j]=0
5. Si A[i][j] almacena un valor no normalizado (se considera que el valor 0 es normalizado) y B[i][j] almacena un valor normalizado, entonces C[i][j]=B[i][j]
6. Si A[i][j] almacena un valor normalizado (se considera que el valor 0 es normalizado) y B[i][j] almacena un valor no normalizado, entonces C[i][j]=A[i][j]
7. En caso contrario, si A[i][j] almacena un valor normalizado y B[i][j] también (se considera que el 0 está normalizado), C[i][j]=A[i][j] + B[i][j]

La matriz D se obtiene de la siguiente forma: D[i][j]= minimo (A[i][j], B[i][j]), donde mínimo es una función que obtiene el valor mínimo de dos valores de tipo *float*. En caso de que alguno de los dos valores sea NaN, el resultado será también NaN. Si un valor es –Inf y el otro +Inf, se entiende que el valor mínimo es –Inf.

Ha de hacerse un adecuado tratamiento de errores, por ejemplo, que los datos almacenados en el fichero no se correspondan con el formato adecuado, que haya menos o más datos de los

indicados según la dimensión de la matriz, etc. El tratamiento consistirá en indicar por pantalla el error detectado y abortar la ejecución del programa.

NOTAS:

- el programa `sumarMin.c` solo podrá hacer uso de las funciones de biblioteca definidas en `stdio.h` y `stdlib.h`.
- A la hora de escribir las matrices en los ficheros C y D, basta con imprimirlo con formato de tipo *float* en cualquier caso, independientemente del valor almacenado en el *float*.
- La lectura y escritura de *floats* en fichero se realiza con las funciones de biblioteca *fscanf* y *fprintf*, cuyo funcionamiento es similar a *scanf* y *printf*.
- La escritura de los valores de tipo *float* al archivo utilizando *fprintf* debe hacerse con el especificador de formato %g.

Ejercicio 2

El objetivo de este ejercicio es realizar el mismo ejercicio anterior, pero en este caso utilizando el ensamblador del MIPS32 y el simulador QTSPIM. En este caso el programa no leerá las matrices de ficheros, sino que su contenido estará indicado en el segmento de datos del programa. Por ejemplo:

```
.data:

   .align 2

   dimension:  .word 3
   matrizA:
             .float 0.0, -2.3, 4.5
             .float N
aN, +Inf, -Inf
             .float 3.4, 6.7, 9.9

   matrizB:
             .float 0.0, -2.3, 4.5
             .float NaN, +Inf, -Inf
             .float 3.4, 6.7, 9.9

   matrizC: .space 36

   matrizD: .space 36

.text:
```

Para realizar este ejercicio tendrá que desarrollarse obligatoriamente, al menos, las dos siguientes subrutinas aparte del `main`:

- `sumarMin`: aceptará como parámetros de entrada la dimensión de las matrices y las direcciones de memoria donde se almacena cada una de las matrices. La función devolverá 0 si no ha habido ningún error en el procesamiento de las matrices y -1 en caso contrario.
- `minimoFloat`: acepta como parámetros de entrada dos valores de tipo *float* y devuelve como resultado un valor de tipo *float*. La función calcula el mínimo de dos valores de acuerdo a la definición de mínimo que se indicó en el ejercicio 1.

Deberá seguirse estrictamente el convenio de paso de parámetros. En caso de que la práctica no incluya al menos estas 2 subrutinas y no se siga el convenio de paso de parámetros, la práctica estará suspensa. Se valorará muy positivamente que la práctica haga uso de más funciones auxiliares.

NOTA: Tenga en cuenta que en el simulador QTSPIM no se pueden utilizar las constantes NaN, +Inf y –Inf, tal y como se han usado en el segmento de datos anterior. En el segmento de datos a utilizar en realidad en la práctica debe cambiarse estas constantes por valores hexadecimal que representen a un valor NaN, +Inf y –Inf.