Práctica 1. Iteración y recursión I (iteración)

En esta práctica y en la siguiente vamos a ver cómo se puede resolver el mismo problema de formas distintas.

El problema elegido es la resolución de ecuaciones por el método numérico de **bisecciones sucesivas** explicado a continuación.

Supongamos que se quiere hallar una de las raíces de la ecuación $f(x) = 0$, donde f es una función continua en el intervalo $[a, b]$ y cuyo valor en los extremos del intervalo $[a, b]$ tiene signos distintos, es decir, que el signo de $f(a)$ es distinto que el de $f(b)$.

Se puede demostrar, aunque resulta obvio, que en el intervalo $[a, b]$ se encuentra, al menos, una de sus raíces, es decir, $\exists x_0 \in [a, b] \mid f(x_0) = 0$.

El algoritmo consiste en ir dividiendo el intervalo en partes iguales de forma que se vaya acotando el lugar donde se encuentra la raíz. Los pasos del algoritmo son los siguientes:

1. Se comprueba que los signos de $f(a)$ y $f(b)$ son diferentes. Si no lo son, el algoritmo termina en error, ya que no hay garantía de que haya una solución en el intervalo $[a, b]$. Si se diera el caso de que en alguno de los extremos $f(x) = 0$, ya habríamos encontrado la solución.

2. Se calculan $p = \dfrac{a + b}{2}$ y $f(p)$. Se pueden dar los siguientes casos:

   a) Si $f(p) = 0$, hemos encontrado la solución y el algoritmo termina, dando como raíz p.

   b) Si el signo de $f(p)$ es el mismo que el de $f(a)$ volveremos al paso 2 con $a = p$.

   c) Si el signo de $f(p)$ es el mismo que el de $f(b)$ volveremos al paso 2 con $b = p$.

Este problema se puede resolver de dos maneras: iterativamente, o sea, mediante un bucle, o recursivamente, es decir, llamando sucesivamente a la misma función.

En esta primera práctica lo resolveremos iterativamente.

EJERCICIO:

1. Construya una función con tres parámetros de punto flotante de simple precisión:

   - `$f12`: comienzo del intervalo donde queremos hallar la raíz.
   - `$f13`: final de dicho intervalo.
   - `$f14`: tolerancia del error en la solución.

   y que devolverá:

   - `$v0`: estado del algoritmo: 0, solución correcta; 1, los extremos del intervalo la función tiene el mismo signo y no hay garantía de encontrar la solución en su interior.
   - `$f0`: raíz de la función $f(x)$, es decir, el valor de x tal que $f(x) = 0$.

   Para llamar a $f(x)$, los alumnos deben llamar a una función en ensamblador definida mediante la etiqueta `Funcion`, la cual recibe la variable independiente, x, a través del registro `$f16`, y retorna el valor $y = f(x)$ en el registro `$f0`.

2. Escriba un programa que, empleando las funciones del ejercicio anterior, pida por teclado los tres parámetros de la función e imprima en la consola la raíz de la ecuación $f(x) = 0$. En caso de que en los extremos del intervalo suministrado la función tenga el mismo signo deberá mostrarse un mensaje de error.

NOTA: El algoritmo teórico descrito más arriba necesita algunos ajustes para su correcta implementación práctica.

EVALUACIÓN Y ENTREGA DE LA PRÁCTICA

La práctica será evaluada a través de la herramienta `Tablon`, disponible `http://tablon-aoc.infor.uva.es/`. Se facilitará un usuario y contraseña a cada grupo de prácticas para acceder al `Tablon` durante la sesión presencial de prácticas.

Para poder evaluar la práctica, el fichero `.asm` enviado al `Tablon` debe cumplir los siguientes requisitos:

- Debe contener únicamente la función iterativa implementada por el grupo. Es decir, no debe contener ni el código correspondiente al `main` ni el procedimiento `Funcion` implementado.
- La función iterativa debe llamarse `Bisec`.
- $f(x)$ debe invocarse mediante una **llamada a un procedimiento** denominado `Funcion`.

Una vez enviado el código, `Tablon` realizará una serie pruebas, testeando diferentes valores de entrada para `Bisec`. Una vez finalizada la batería de pruebas, se podrá consultar el `Tablon` el número de pruebas superadas y el porcentaje de pruebas completado, además de otra información relevante, como el número de instrucciones ejecutadas, el número de líneas de código de la función `Bisec` y el número de excepciones en tiempo de ejecución. Estas métricas sirven para elaborar el ranking y desempatar en caso de empate en el número de casos de prueba superados. Sin embargo, **la calificación de la práctica dependiente del `Tablon` dependerá exclusivamente del número pruebas superadas**. El resto de métricas y el ranking no se tendrá en cuenta para la evaluación.

Finalmente, se debe entregar en el Aula Virtual el fichero `.asm` que obtuviera el mejor resultado en el ranking de la herramienta `Tablon`. El nombre del fichero para el Aula Virtual será el número de *request* del `Tablon` con 5 cifras y la extensión `.asm`

NOTA: La calificación no depende exclusivamente del resultado del `Tablon`. El código también se revisará manualmente para evaluar su calidad y proporcionar *feedback* a los alumnos en las siguientes prácticas.

Fecha límite de entrega en el Aula Virtual: 3 de octubre a las 23:55 h.

No se evaluarán las entregas del `Tablon` que no se hayan depositado dentro del plazo en el Aula Virtual.

Notas sobre los cálculos en punto flotante

El coprocesador de punto flotante de MIPS, c1, dispone de 32 registros de punto flotante en simple precisión (32 bits) (`$f0` a `$f31`). Estos registros no tienen propósitos específicos como los enteros.

Para realizar comparaciones el coprocesador c1 tiene 8 banderas de condición, o *flags*, que se activan con las instrucciones `c.**.s` $cc\ fs,\ ft$ (para comparaciones de simple precisión) donde ** puede ser: "`eq`" (igual), "`lt`" (menor) o "`le`" (menor o igual) y cc es el número de una de las 8 banderas de condición (de 0 a 7); si no se especifica cc se supone que es 0.

Las bifurcaciones se efectúan mediante las instrucciones `bc1t` $cc\ Etiqueta$ o `bc1f` $cc\ Etiqueta$, que bifurcan a $Etiqueta$ si el *flag* cc está a 1 o 0 respectivamente.

Los *flags* solo cambian por medio de las instrucciones `c.**.s` y no se modifican por otros motivos.

Ejemplo 1:

Supongamos que queremos bifurcar a `Bucle` si el contenido del registro `$f0` es menor o igual que el de `$f1`. Las instrucciones necesarias para hacerlo serían las siguientes:

```
c.le.s 1 $f0, $f1
bc1t   1 Bucle
```

Si quisiéramos bifurcar cuando `$f0` fuera mayor que `$f1` emplearíamos `bc1f 1 Bucle`, en vez de `bc1t`. Si en lugar de emplear el *flag* 1, empleáramos el 0, no sería necesario mencionarlo y las instrucciones quedarían así:

```
c.le.s $f0, $f1
bc1t   Bucle
```

Una cuestión interesante es que las instrucciones de comparación y bifurcación no tienen que estar necesariamente seguidas. Así el mismo *flag* puede emplearse para varias bifurcaciones.

También es importante recordar que no hay instrucciones de punto flotante que trabajen con operandos inmediatos, por lo que las constantes hay que declararlas en memoria. Por otra parte, los apuntadores de las instrucciones de punto flotante son registros enteros, ya que las direcciones siempre son enteras.

Ejemplo 2:

Para cargar las constantes 0 y 5 en los registros de punto flotante `$f2` y `$f5` en simple precisión emplearemos las siguientes instrucciones:

```
.data
Cero:  .float 0.0
Cinco: .float 5.0
...
.text
...
la     $t0, Cero
lwc1   $f2, 0($t0)
la     $t0, Cinco
lwc1   $f5, 0($t0)
```

Las dos últimas instrucciones se podrían sustituir por `lwc1 $f5, 4($t0)`, ya que los números de punto flotante almacenados en simple precisión ocupan 4 bytes.