HOJA DE PROBLEMAS DEL TEMA 4.
PROGRAMACIÓN EN ENSAMBLADOR DE MIPS
NÚMEROS REALES

---

1. Sea una función ***f(x)*** y sean dos puntos de la misma con coordenadas ***(x₁,y₁)*** y ***(x₂,y₂)***. El valor de la función en un cierto punto ***xₚ*** tal que ***x₁<xₚ<x₂*** se puede interpolar mediante la fórmula siguiente:

$$f(x_p) = \frac{(x_p - x_1) \cdot (y_2 - y_1)}{x_2 - x_1} + y_1$$

Se pide realizar un programa en ensamblador que, dadas las coordenadas cartesianas de sendos puntos pertenecientes a una función ***f(x)***, calcule el valor de la función para un cierto punto intermedio de coordenada ***xₚ*** dada.

A continuación se muestra una posible solución en C:

```
float x1, y1, x2, y2, xp, yp;
int main (void) {
     yp = ((xp-x1) * (y2-y1)) / (x2-x1) + y1;
     return 0;
}
```

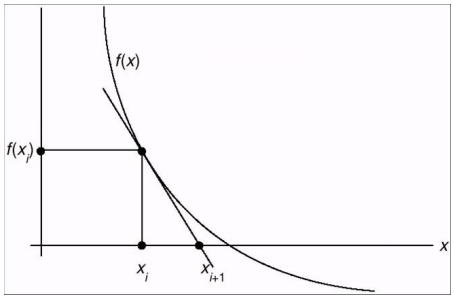
Las variables ***x1***, ***x2***, ***y1***, ***y2***, ***xp*** e ***yp*** deben existir en la solución con el mismo nombre, y tendrán un valor inicial desconocido. Para realizar las pruebas se introducirán manualmente los valores necesarios en las variables. Escribir la solución en un fichero llamado **interpolacion_lineal_float.asm**.

---

2. El método de Newton-Raphson sirve para calcular las raíces de una ecuación $f(x) = 0$ de forma iterativa. La fórmula de recurrencia se obtiene a partir de la definición de derivada de una función:

$$f'(x_i) = \frac{f(x) - f(x_i)}{x - x_i}$$

De aquí se obtiene la ecuación de la recta tangente a $f(x)$ en $x = x_i$:



$$t(x) = f'(x_i) \cdot (x - x_i) + f(x_i)$$

Como se ve en la figura, la recta tangente corta el eje Y en el punto $x = x_{i+1}$ donde

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Esta ecuación es la base del método. Para calcular la solución de una función $f(x)$ se toma un valor semilla $x = x_0$ próximo a la raíz buscada como primera aproximación a la solución. Después se aplica la ecuación anterior para obtener una nueva aproximación $x = x_1$. El proceso se repite hasta que el método converja a una solución $x = x_{i+1}$, lo que se detecta cuando $|x_{i+1} - x_i| < \varepsilon$ donde ε es un valor positivo y muy próximo a 0 establecido a priori en función de la precisión buscada.

Por ejemplo, para calcular la raíz cuadrada de un número Z mediante el método de Newton-Raphson se puede partir de la función

$$f(x) = x^2 - Z$$

cuya solución es

$$x = \sqrt{Z}$$

La derivada de la función es

$$f'(x) = 2x$$

Sustituyendo $f(x)$ y $f'(x)$ en la ecuación del método se obtiene la ecuación de recurrencia

$$x_{i+1} = 0{,}5 \cdot \left(x_i + \frac{Z}{x_i}\right)$$

Se desea diseñar en ensamblador de MIPS un programa que calcule la raíz cuadrada de un número Z mediante el método de Newton-Raphson, utilizando la ecuación de recurrencia anterior. Una posible solución en C es:

```
float z, raiz_z, epsilon, error;
register float xi, xi_1;
int main (void) {
     if (z < 0.0)
          return 0;
     xi_1 = 1; // Semilla si z > 0
     do {
          xi = xi_1;
          xi_1 = 0.5*(xi-z/xi);
     ] while (fabs(xi_1-xi) >= epsilon);
     raiz_z = xi_1;
     error = fabs(raíz_z-sqrt(Z));
     return 0;
}
```

El código anterior calcula además el error absoluto cometido comparando el resultado obtenido frente al de la raíz cuadrada calculada con **sqrt**. Las variables ***z, raíz_z, espsilon*** y ***error*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Escribir la solución en el fichero **sqrt.asm**.

---

3. Diseñar un programa en ensamblador de MIPS que calcule el valor de un polinomio $P(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \cdots + a_1 \cdot x + a_0$ para un valor concreto $x = x_0$ mediante el método de Horner, que realiza dicho cálculo del siguiente modo:

$$P(x_0) = a_0 + x_0 \cdot (a_1 + x_0 \cdot (a_2 + \cdots + x_0 \cdot (a_{n-1} + x_0 \cdot a_n) \ldots))$$

Se puede partir de la siguiente versión en C:

```
int grado;
float polinomio[10], x0, valor;
register int i;
int main (void) {
     valor = polinomio[grado];
     i = grado - 1;
     while (i >= 0) {
          valor = valor * x0 + polinomio[i];
          i--;
     }
     return(0);
}
```

El grado máximo del polinomio considerado es 9. Las variables ***grado***, ***polinomio***, ***x0*** y ***valor*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Excepcionalmente, para realizar las pruebas se podrá dar un contenido inicial conocido a los coeficientes del polinomio y a la variable ***grado***, si bien el valor inicial de ***x0*** se introducirá manualmente. Escribir la solución en el fichero **horner.asm**.

---

4. Diseñar un programa en ensamblador de MIPS que compruebe si se cumple la propiedad asociativa de la suma con tres números dados en coma flotante de precisión simple. Se puede partir de la siguiente versión en C:

```
float x, y, z;
register float e1, e2;
int iguales;
int main (void) {
     e1 = (x+y)+z;
     e2 = x+(y+z);
     iguales = (e1 == e2);
     return(0);
}
```

Las variables ***x***, ***y***, ***z*** e ***iguales*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Escribir la solución en el fichero **asociativa_float.asm**. Probar el programa con los juegos de ensayo de la tabla y rellenar los huecos vacíos:

Caso	x	y	z	(x+y)+z	x+(y+z)	¿Iguales?
1	3.0	2.0	4.0			
2	1.0	1,00E+08	1,00E-08			
3	1,00E-06	5.0	-5.0			
4	1,00E-10	1,00E+04	-1,00E+04			
5	1,00E+07	1.0	1,00E-07			

Dar una explicación razonada a los resultados obtenidos.

---

5. Diseñar un programa en ensamblador de MIPS equivalente al siguiente programa escrito en C:

```
float x;
register float x_1;
int iguales;
int main (void) {
     x_1 = x + 1.0;
     iguales = (x == x_1);
     return(0);
}
```

Las variables ***x*** y ***x_1*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Escribir la solución en el fichero **suma1_float.asm**. Probar el programa con los juegos de ensayo de la tabla y rellenar los huecos vacíos:

Caso	x	x+1	¿x = x+1?
1	3.0		
2	-1.0		
3	-1,00E+03		
4	1,00E+08		
5	1,00E+10		

Dar una explicación razonada a los resultados obtenidos.

---

6. Diseñar un programa en ensamblador de MIPS que compruebe si es posible calcular correctamente el cuadrado de un número dado en coma flotante de precisión simple. Es decir que, dados como datos de entrada un número ***x*** y su cuadrado, hay que comprobar si ***cuadrado = x · x***. Se puede partir de la siguiente versión en C:

```
float x, cuadrado;
register float x2;
int iguales;
int main (void) {
     x2 = x * x;
     iguales = (cuadrado == x2);
     return(0);
}
```

Las variables ***x*** y ***cuadrado*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Escribir la solución en el fichero **cuadrado_float.asm**. Probar el programa con los juegos de ensayo de la tabla y rellenar los huecos vacíos:

Caso	x	cuadrado	x*x	¿Iguales?
1	2.0	4.0		
2	0.1	0.01		
3	0.2	0.04		
4	0.5	0.25		

Dar una explicación razonada a los resultados obtenidos.