HOJA DE PROBLEMAS DEL TEMA 4.
PROGRAMACIÓN EN ENSAMBLADOR DE MIPS
SUBRUTINAS

---

1. Se pretende desarrollar un programa que calcule el factorial de un número positivo. Para ello se escribirá una subrutina llamada *factorial* que realizará el cálculo, y un programa principal en el que se creará espacio para las variables y que incluirá una llamada a la subrutina. Las características de la subrutina *factorial* son las siguientes:

- Recibe su único parámetro de entrada en el registro de argumento **$a0**.
- Devuelve el resultado en el registro **$v0**.
- Es una subrutina "hoja".

Se puede partir de la siguiente versión en C:

```
int n, v;

int main (void) {
     if (n >= 0) {
          v = factorial(n);
          return 0;
     }
     else
          return -1;
}

int factorial (int n) {
     int i, f;
     f = 1;
     i = 1;
     while (i < n) {
          i = i + 1;
          f = f * i;
     }
     return f;
}
```

Las variables ***n*** y ***v*** 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. **Se deberá respetar el convenio de uso de registros en subrutinas**. Para escribir la solución se creará un directorio aparte, y dentro de él se escribirá el programa principal en un fichero llamado **main_factorial.asm**, y la subrutina *factorial* en otro fichero llamado **factorial.asm**.

---

2. Repetir el ejercicio anterior, pero ahora la subrutina *factorial* deberá ser recursiva. Se deberá crear un directorio independiente para la solución, y en él se copiará el fichero **main_factorial.asm** desarrollado para el ejercicio previo, y se escribirá la subrutina de

cálculo del factorial en un nuevo fichero llamado **factorial_recursivo.asm**. Se puede partir de la siguiente versión de la subrutina en C:

```
int factorial (int n) {
     if (n == 0)
          return 1;
     else
          return (n * factorial (n-1));
}
```

Es preciso tener en cuenta que ahora la subrutina *factorial* es una subrutina "tallo", y que **en la solución se deberá respetar el convenio de uso de registros en subrutinas.**

---

3. Se pretende diseñar un programa en ensamblador de MIPS que calcule el valor de un número combinatorio. Se recuerda que un número combinatorio se expresa así

$$\binom{n}{k}$$

y que su valor se calcula como

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

Para ello nos apoyaremos sobre la subrutina *factorial* de los ejercicios anteriores. Se puede partir de la siguiente versión en C (no se incluye la subrutina de cálculo del factorial):

```
int n, k, r;

int main (void) {
     if ( (k >= 0) && (n >= k) )
          r = factorial(n) / (factorial(k)*factorial(n-k));
          return 0;
     }
     else
          return -1;
}
```

Las variables ***n***, ***k*** y ***r*** 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. **Se deberá respetar el convenio de uso de registros en subrutinas**. Para escribir la solución se creará un directorio aparte, y dentro de él se escribirá el programa principal en un fichero llamado **main_combinatorio.asm**, y se copiará en el mismo directorio el fichero con la subrutina *factorial* de ejercicios anteriores (puede ser la versión iterativa o la recursiva).

---

4. Se desea diseñar en ensamblador de MIPS un programa que calcule la media aritmética de un vector de números en coma flotante de precisión simple. Se utilizará una subrutina llamada ***promedio***, que recibirá como parámetros un vector de datos de tipo **float** y un entero con el número de elementos del mismo. Las características de la subrutina ***promedio*** son las siguientes:

- Su primer parámetro es el vector pasado por referencia, y lo recibe en el registro de argumento **$a0**.
- Su segundo parámetro es el número de elementos del vector, y lo recibe en el registro de argumento **$a1**.
- Devuelve el resultado en el registro **$f0**.
- Es una subrutina "hoja".

Se podrá partir de la siguiente versión del programa en C:

```
#define N 10
float vector[N];
float media;

int main (void) {
      media = promedio(vector,N);
      return 0;
}

float promedio (float *vec, int n) {
      int i;
      float v;
      i = 0;
      v = 0.0;
      do {
            v = v + vec[i];
            i++;
      } while (i < n);
      return (v/n);
}
```

Las variables ***vector, n*** y ***media*** 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 elementos del vector. **Se deberá respetar el convenio de uso de registros en subrutinas**. Escribir la solución, incluyendo el programa principal y la subrutina, en el fichero **promedio_float.asm**.

---

5. Realizar un programa en ensamblador de MIPS que ordene tres números enteros de mayor a menor. Para ello se diseñará una subrutina auxiliar que realice la tarea antedicha. La subrutina se denominará ***ordenar3***. Los tres parámetros de la subrutina se pasarán ***por referencia*** a través de los registros **$a0**, **$a1** y **$a2**. La subrutina es una subrutina "hoja", y pondrá el número más grande en la variable pasada por referencia en primer lugar (apuntada por **$a0**), el número intermedio en la variable pasada por referencia en segundo lugar (apuntada por **$a1**), y el número menor en la variable pasada por referencia en tercer lugar (apuntada por **$a2**). Se podrá partir de la siguiente versión del programa en C:

```
int x,y,z;
int main (void) {
      ordenar3(&x,&y,&z);
      return 0;
}
```

```
void ordenar3 (int *x, int *y, int *z) {
	int tmp;
	if (*x < *y) {
		tmp = *x;
		*x = *y;
		*y = tmp;
	}
	if (*x < *z) {
		tmp = *x;
		*x = *z;
		*z = tmp;
	}
	if (*y < *z) {
		tmp = *y;
		*y = *z;
		*z = tmp;
	}
	return;
}
```

Las variables ***x, y*** y ***z*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Para realizar las pruebas se introducirán manualmente los valores necesarios en las variables. **Se deberá respetar el convenio de uso de registros en subrutinas**. Escribir la solución, incluyendo el programa principal y la subrutina, en el fichero **ordenar_int.asm**.