# Práctica 8
# Coma flotante 1

## Objetivos

- Recordar y afianzar la representación de números en coma flotante según el estándar IEEE 754
- Conocer la Unidad de Coma Flotante del MIPS.
- Saber escribir programas de ensamblador utilizando instrucciones de coma flotante del MIPS

## Material

Simulador MARS y un código de partida

## Teoría

### Representación de la coma flotante

En la mayoría de los computadoras los números reales se representan de forma binaria utilizando el formato estandarizado IEEE 754 (Institute of Electrical and Electronic Engineers Standard 754). Hay tres formatos básicos de distinto tamaño: simple precisión (32 bits), doble precisión (64 bits) y doble precisión extendida (128 bits). Los bits están divididos en tres campos de tamaño fijo: el primero (S) es el signo de tamaño 1 bit, el segundo es el exponente (E) y el tercero es la mantisa (M). La diferencia entre los tres formatos está en el número de bits de los campos exponente y mantisa. Así, en simple precisión el exponente es de 8 bits, doble precisión de 11 bits y doble precisión extendida de 15 bits. La mantisa tiene 23 bits en simple precisión, 52 en doble precisión y 112 en doble precisión extendida.

El bit de signo, S, indica el signo del número, 0 positivo y 1 negativo. El exponente E es positivo y sigue una notación sesgada para incluir el signo, está representado en exceso $2^{q-1}$-1 donde q son los números de bits del exponente. En simple precisión el exceso es 127 (q=8).

La mantisa está normalizada con un 1 implícito a la izquierda de la coma decimal. Por lo tanto los valores representados de la mantisa están comprendidos entre 1,0000... y 1,1111...

El valor 0 y el valor $2^q$-1 (todo a unos) para el exponente están reservados para los casos especiales que se muestran en la tabla 1:

| Exponente | Mantisa | Valor |
|---|---|---|
| **$2^q$-1** | ≠0 | NaN (Not a Number) |
| **$2^q$-1** | 0 | +∞ i -∞ dependiendo del signo S |
| **0** | 0 | +0 i -0 dependiendo del signo S |
| **0** | ≠0 | Nombres desnormalizados ($0.M \times 2^{-126}$) |

*Tabla 1: Casos especiales del Formato IEEE 754.*

La representación del formato de un valor en coma flotante en simple precisión de 32 bits se muestra en la figura 1:

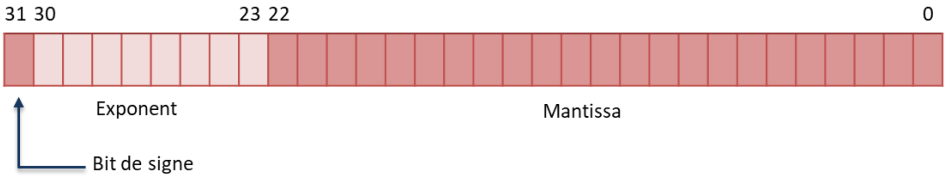


*Figura 1: Formato IEEE 754 en simple precisión*

El valor representado de un número en coma flotante en simple precisión es:

$$N = (-1)^S \times 1.M \times 2^{E-127}$$

donde S es el signo y E el exponente

## La coma flotante en MIPS

El procesador MIPS opera con los números enteros en la unidad central de procesamiento (CPU). En cambio, las operaciones de coma flotante se hacen en una unidad separada de coma flotante, la FPU (Floating Point Unit) que en MIPS recibe el nombre de coprocesador 1. Este coprocesador opera con números representados en coma flotante en simple precisión (32 bits) y doble precisión (64 bits).

La FPU del MIPS tiene un conjunto especial de 32 registros de 32 bits numerados del $f0 al $f31. Cada uno de ellos puede almacenar números en coma flotante de simple precisión. Para almacenar números de doble precisión de 64 bits se necesitan parejas de estos registros. Esto significa que sólo podrán utilizarse registros de coma flotante con numeración pareja para referirse a ellos. Por ejemplo, cuando se almacena un valor de doble precisión en lo registro $f0 realmente está guardándose en la pareja de registros $f0 y $f1.
Cada una de las unidades (CPU y FPU) tiene sus propias instrucciones aritméticas para operar con cada tipo de datos y sus propias instrucciones de acceso a memoria. Además, existen una serie de instrucciones de transferencia para mover datos entre los registros de las dos unidades. En la figura 2 se muestra el modelo del procesador MIPS.

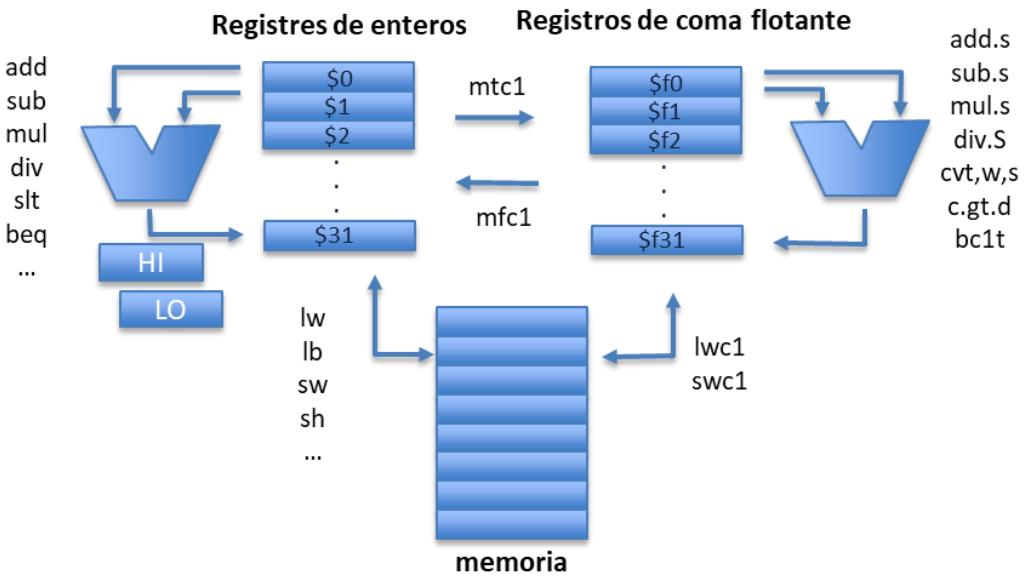


*Figura 2. Modelo del procesador MIPS*

Además, el coprocesador de coma flotante tiene ocho indicadores de código de condición (cc) numerados del 0 al 7. Las instrucciones de comparación los utilizan para guardar el resultado de las comparaciones y las instrucciones de salto y movimiento condicional para decidir si hacen o no la operación.
Por cuestiones de simplicidad utilizaremos sólo números en coma flotante representados en simple precisión (32 bits) aunque indicaremos también las instrucciones en doble precisión.

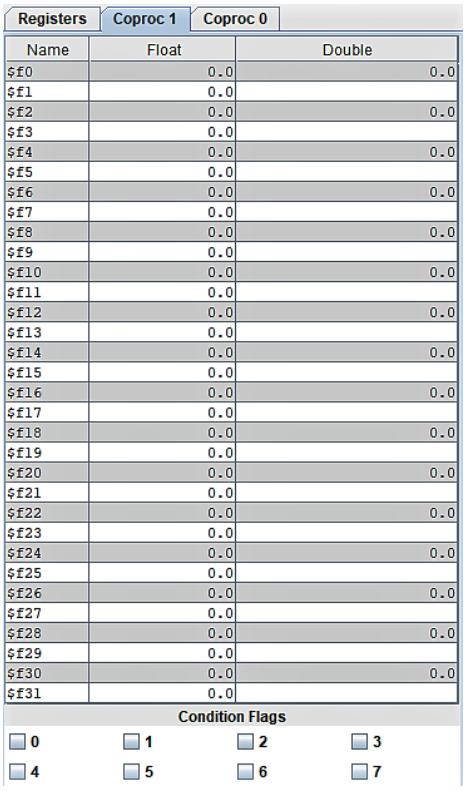


*Figura 3 . Registros mostrados por el MARS junto a los códigos de condición*

Estos registros junto con los códigos de condición pueden verse en el MARS seleccionando la pestaña *Coproc* 1 como muestra la figura 3:

También, en el caso de los registros en coma flotante (de manera similar a los registros de la CPU), se utiliza un convenio de uso, no seguirlo podría traer problemas en el paso de parámetros, obtención de resultados de funciones y utilización de los registros a través de las llamadas a subrutinas. En la tabla 2 se muestra el convenio de utilización de los registros de coma flotante del MIPS.

| Registro | Utilización |
|---|---|
| **$f0..$f3** | Utilizado para contener los resultado de las funciones de tipos coma flotante |
| **$f4..$f11** | Registros temporales utilizados para evaluar expresiones, sus valores NO están preservados por las llamadas a subrutinas. |
| **$f12 .. $f15** | Utilizados para pasar parámetros a subrutinas , sus valores NO están preservados a través de las llamadas a subrutinas |
| **$f16..$f19** | Más registros temporales. NO preservados a través de llamadas a subrutinas |
| **$f20..$f30** | Registros de guardado, sus valores están preservados a través de las llamadas a subrutinas. |

*Tabla 2. Convenio de utilización de los registros en coma flotante del MIPS.*

El simulador MARES tiene una herramienta de representación de la coma flotante que ilustra los números en coma flotante de simple precisión. Ved `Tools-> Floating Point Representation` y abrid la ventana, veréis lo que muestra la figura 4.

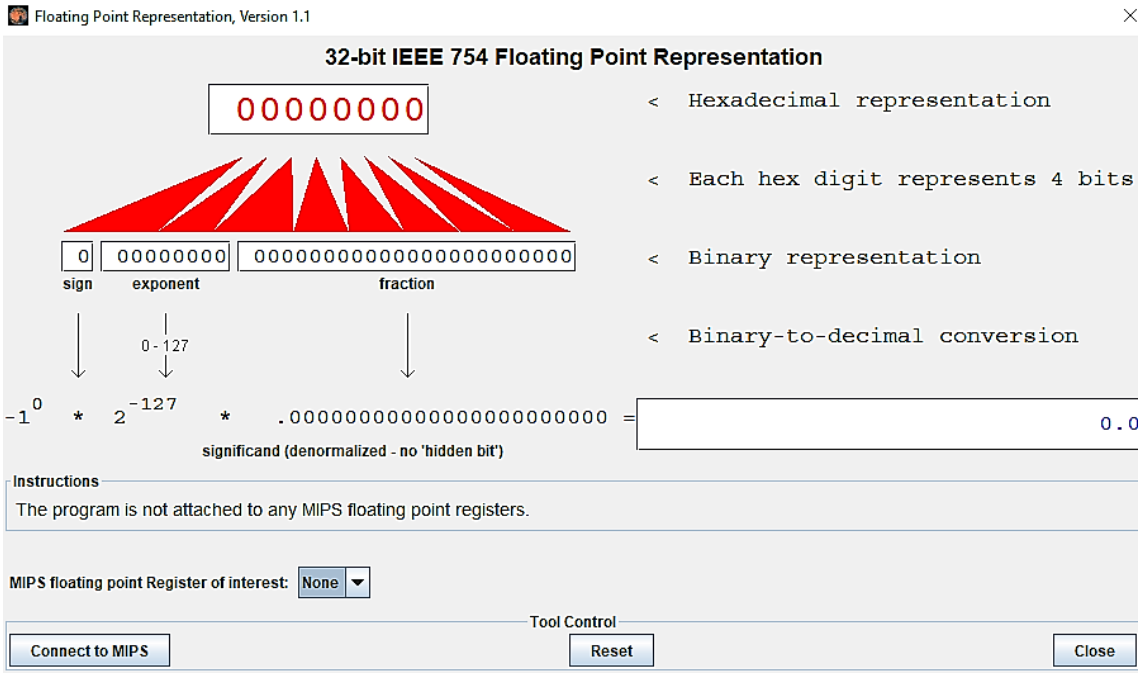


*Figura 4. Herramienta de representación en coma flotante del MARS*

Se puede hacer uso de esta herramienta para comprobar el valor decimal y binario de los números en coma flotante.

- Comprobad que el número decimal de C0E00000 es -7.0 y que la representación en 32 bits de simple precisión de 4.25 es 40880000

# Desarrollo de la práctica. Conjunto de instrucciones

## 1. Operaciones aritméticas

Como la FPU es una unidad separada, las operaciones en coma flotante sólo usan los propios registros de la unidad. Las instrucciones MIPS pera sumar y restar números en coma flotante de simple precisión son:

```
add.s $f1, $f0, $f1 # Suma en simple precisión, $f1← $f0+$f1
sub.s $f0, $f0, $f2 # Resta en simple precisión, $f0← $f0-$f2
```

En el último ejemplo, el registro $f0 contendrán el resultado de la resta. A diferencia del registro $0 del banco de registros de la CPU, la FPU no tiene ningún registro con valor 0. De manera similar, excepto porque no se pueden utilizar registros con numeración impar, podemos operar con los números en coma flotante de doble precisión con las siguientes instrucciones:

```
add.d $f2, $f4, $f6 #suma en doble precisión, {$f2,$f3} ←{$f4,$f5}+{$f6,$f7}
sub.d $f0, $f2, $f4 #resta en doble precisión, {$f0,$f1} ←{$f2,$f3}+{$f4,$f5}
```

Fijaos que sólo se indica en las instrucciones de doble precisión el registro que contiene los 32 bits de mayor peso del número que es el que tiene numeración par.
Se indica que es una instrucción de simple precisión con la extensión *.s* y una de doble precisión con la extensión *.d*.

Las siguientes instrucciones son sintácticamente incorrectas, no están permitidas:

```
add.s $s0, $s0, $s1        #No está permitida puesto que add.s espera registros FPU
add   $f0, $f2, $f2        #No permitida porque add espera registros CPU

add.d $f0, $f2, $f5        #No está permitida porque f5 es impar
addi.s $f0, $f1, 2.3       #No  han instrucciones inmediatas en coma flotante
```

Para la multiplicación y la división se tienen las siguientes instrucciones en simple precisión (extensión *.s*) y doble precisión (extensión *.d*):

```
mulo.s $f0, $f1, $f2 #Multiplicación en simple precisión
div.s $f0, $f1, $f2 #División en simple precisión

mulo.d $f0, $f0, $f2 #Multiplicación en doble precisión,
div.d $f0, $f0, $f2 #División en doble precisión
```

- ¿Cuál es la razón por la que no hay instrucciones aritméticas en coma flotante con datos inmediatos?
- ¿Por qué no hay registros HI y LO para guardar el resultado de la multiplicación y división en coma flotante del mismo modo que con los números enteros?.

Todas las instrucciones aritméticas de coma flotante siguen un formato de instrucción tipo R con el mismo código de operación: 0x11. En la tabla 3 se resumen las instrucciones aritméticas.

| Simple precisión | Acción | Doble precisión |
|---|---|---|
| add.s $f0, $f1, $f2 | $f0←$f1+$f2 | add.d $f0, $f2, $f4 |
| sub.s $f0, $f1, $f2 | $f0←$f1-$f2 | sub.d $f0, $f2, $f4 |
| mulo.s $f0, $f1, $f2 | $f0←$f1*$f2 | mulo.d $f0, $f2, $f4 |
| div.s $f0, $f1, $f2 | $f0←$f1/$f2 | div.d $f0, $f2, $f4 |
| sqrt.s $f0, $f1 | $f0←Square root($f2) | sqrt.d $f0, $f2 |
| neg.s $f0, $f1 | $f0←Neg($f2) | neg.d $f0, $f2 |
| abs.s $f0, $f1 | $f0←Abs($f2) | abs.d $f0, $f2 |

*Tabla 3. Instrucciones aritméticas en coma flotante*

## 2. Operaciones de movimiento de datos

Para poder hacer cualquier tipo de operación aritmética se tienen que rellenar los registros FPU y después de hacerse las operaciones el resultado se tiene que guardar en otro registro FPU. Hay dos formas de mover datos desde o hacia a los registros de coma flotante, bien mediante registros bien mediante la memoria.
La primera forma es mover palabras desde o hacia los registros CPU. Esta operación la realizan las instrucciones:

```
mtc1   $s0, $f0  # “move to coprocessor 1”, $f0 ← $s0
mfc1   $s0, $f0  # “move from coprocessor 1”, $s0 ← $f0
```

Fijaos en el orden de los operandos de estas instrucciones.

El MIPS también permite mover datos entre los mismos registros de la FPU con la instrucción `move`:

```
mov.s $f0, $f1      #copia $f1 en $f0.
mov.d $f4, $f2      #copia $f2-$f3 en $f4-$f5.
```

En la tabla 4 se recogen las instrucciones de movimiento de datos entre registros.

| Simple precisión | Acción | Doble precisión |
|---|---|---|
| mov.s $f0, $f1 | $f0 ← $f1 | mov.d |
| mfc1 $t0, $f0 | $t0 ← $f0 | |
| mtc1 $t0, $f0 | $f0 ← $t0 | |

*Tabla 4. Instrucciones movimiento de datos en coma flotante*

Código ejemplo 1

```
                # Código ejemplo 1.

li $t0, 0xFF800000      # Menos infinito
mtc1 $t0, $f12          # movemos $t0 -> $f12=0xFF800000
li $t1, 0x7F8003A0      # Not a Number  (NaN)
mtc1 $t1, $f20          # movemos $t1 -> $f20=0x7F8003A0
```

El código mete un valor en hexadecimal en $t0 y en $t1. Según el estándar IEEE 754 el valor en $t0 representa el -∞ y el que hay en $t1 representa un NaN. El programa pasa estos valores a los registros $f12 y $f20 de la FPU.

- Haciendo uso de la herramienta de representación en coma flotante del MARS, comprueba que realmente en $f12 hay un -∞ y en $f20 un *NaN* según el estándar IEEE 754.
- Haz que el contenido de $f1 sea el valor 1 en coma flotante y el de $f2 el valor -2.5 en coma flotante.
- Di una manera de escribir un 0.0 en $f0 con sólo una instrucción máquina.

## 3. Conversión de tipo

Si nos interesa hacer una operación con dos operandos fuente (suma, resto, multiplicación...) los dos tienen que ser del mismo tipo. Si uno de ellos es un valor entero y el otro un valor en coma flotante entonces uno de los operandos se tiene que convertir al tipo del otro, puesto que la operación se tendrá que hacer o con los circuitos de coma flotante o con los circuitos de enteros. En cualquier caso la conversión se hará en la FPU, tanto si es de flotante a entero cómo si es de entero a flotante.
La instrucción MIPS que hace la conversión es `cvt._._` dónde lo subrayado se rellena con el símbolo *s* (para coma flotante de simple precisión), con *d* (para coma flotante de doble precisión), o con *w* (para un entero). Por ejemplo:

```
cvt.s.w $f0, $f0
```

coge los 32 bits que hay en $f0 y que representan un valor entero y los convierte en un número en coma flotante del mismo valor. El valor resultante lo guarda en $f0.

En la tabla 5 se recogen las instrucciones de conversión de tipo del MIPS.

| Simple precisión | Acción | Doble precisión |
|---|---|---|
| cvt.s.w $f0,$f2 | $f0←Conversión a simple precisión del entero en $f2 | cvt.s.d |
| | | cvt.d.w |
| | | cvt.d.s |
| cvt.w.s $f0,$f2 | $f0←Conversión a entero de $f2 | cvt.w.d |
| ceil.w.s $f0,$f2 | $f0 ←Asigna el menor número entero igual o mayor que $f2 | ceil.w.d |
| floor.w.s $f0,$f2 | $f0 ←Asigna el mayor número entero igual o menor que $f2 | floor.w.d |
| trunc.w.s $f0,$f2 | $f0 ←Asigna el entero truncado de $f2. | trunc.w.d |

*Mesa 5. Instrucciones de conversión de tipo en coma flotante*

Código ejemplo 2

```
                # Código ejemplo 2. Conversión de tipo


addi $s0, $0, -8        # Metemos $s0 = 0xfffffff8
mtc1 $s0, $f0           # movemos $s0 -> $f0 = 0xfffffff8
cvt.s.w $f0, $f0        # de w a s -> $f0 = 0xc1000000
```

El código mete el valor -8 en $s0 y lo pasa al registro $f0 de la FPU, si no lo convertimos el dato que habría en $f0 no correspondería al valor -8.

- Haciendo uso de la herramienta de representación en coma flotante del MARS, comprueba que realmente en $f0 después de la conversión hay un -8.
- ¿Qué valor consideraría la máquina que habría en $f0 si no hiciésemos la conversión con la instrucción `cvt.s.w`?. Aprovecha que puedes ver contenidos en decimal y en hexadecimal.
- Haz que el contenido de $f1 sea el valor 1 en coma flotante y el de $f2 el valor -2 en coma flotante utilizando las instrucciones de conversión de tipo.

Código ejemplo 3

```
                # Código ejemplo 3

li $s0, 841242345
mtc1 $s0, $f0           # movemos de $s0 a $f0
cvt.s.w $f1, $f0        # Conversión de entero-simple precisión
cvt.w.s $f1, $f1        # Conversión de simple precisión-entero
mfc1 $s1, $f1           # movemos de $f1 a $s1
```

- Ensambla el código y comprueba el resultado.
- ¿Cuál es la razón por la que al finalizar el programa los contenidos de $s0 y $s1 son distintos?

Código ejemplo 4

```
                # Código ejemplo4 Desbordamiento

addi $s0, $0, 1
sll $s0, $s0, 30            # $s0 = 2^30
mtc1 $s0, $f0
cvt.s.w $f0, $f0            # $f0 = 2^30
mul.s $f0, $f0, $f0        # $f0 = 2^60
mul.s $f0, $f0, $f0        # $f0 = 2^120
mul.s $f0, $f0, $f0        # $f0 = 2^240 -> overflow
```

- Ensambla y ejecuta el código. ¿Qué valor representa en el formato IEEE 754 el contenido final de $f0?
- Observa que no ha ocurrido ninguna excepción a la ejecutar el código. ¿Ocurre lo mismo al dividir por cero? ¿Y al hacer 0/0? Añade instrucciones al código anterior y haz la prueba.

Código ejemplo 5

```
# detectar casos especiales del formato IEEE 754

.data
   mmask: .word 0x007FFFFF
   emask: .word 0x7F800000
   exp1: .word 255

 .text
            addi $s0, $0, 1
            sll $s0, $s0, 30 # $s0 = 2^30
            mtc1 $s0, $f0
            cvt.s.w $f0, $f0 # $f0 = 2^30
            mul.s $f0, $f0, $f0 # $f0 = 2^60
            mul.s $f0, $f0, $f0 # $f0 = 2^120
            mul.s $f0, $f0, $f0 # $f0 = 2^240 -> overflow

            #Valor a comprobar en $f0

            mfc1 $s0,$f0
            lw $t4,mmask # cargar mascara de la mantisas
            and $t0,$s0,$t4 # extraer mantisa de $s0
            lw $t4,emask # cargar mascara del exponente
            and $t2,$s0,$t4 # extraer exponente de $s0
            srl $t2,$t2,23 # desplazar exponente

            lw $t3,exp1   #cargamos valor exponente todo a unos
            beq $t2,$t3,exp_a_1 #exponente todo a unos?
```

Se ha añadido al ejemplo 4 un fragmento de código que extrae el exponente y la mantisa de un número representado en formato IEEE 754 para detectar si se ha producido desbordamiento y en ese caso mostrar un mensaje de aviso en la consola. El código está incompleto.

- Analiza el código y observa cómo se extrae el exponente y la mantisa.
- Completa el código para que muestre en consola un mensaje de error por desbordamiento. Comprueba que funciona correctamente.
- Completa el código para que detecte todos los casos especiales y muestre mensajes en la consola. Haz distintas pruebas y comprueba que funciona.

## 4. Llamadas al sistema

Se puede leer un número en coma flotante de teclado y escribirlo en consola utilizando la instrucción *syscall* pasando el número del servicio en $v0 como se muestra en la tabla 8:

| Servicio | Código de llamamiento. | Argumentos | Resultado |
|---|---|---|---|
| **Print Float** | 2 | $f12=Flotante en simple precisión | Imprimir en consola el contenido de $f12 |
| **Print Double** | 3 | $f12=Flotante en doble precisión | Imprimir en consola el contenido de $f12 |
| **Read Float** | 6 | | Lee de la consola un flotante en simple precisión y lo guarda en $f0 |
| **Read Double** | 7 | | Lee de la consola un flotante en simple precisión y lo guarda en $f0 |

*Tabla 8. Códigos de servicio de syscall para coma flotante*

## Ejercicios a entregar

- Completa el siguiente código de partida que pide el radio por teclado y tiene que calcular y mostrar en la consola la longitud de la circunferencia y el área del círculo.

```
.data
demanaPi :  .asciiz "Dame el valor de pi..."
pideRadio:  .asciiz "Dame el radio... "
long:       .asciiz "Longitud de la circunferència = "
super:      .asciiz "Área del círculo = "

.text
            li $v0,4
            la $a0,demanaPi
            syscall
            li $v0,6
            syscall
            mov.s $f1, $f0
            li $v0,4
            la $a0,pideRadio
            syscall
```

```
            li $v0,6
            syscall
            li $v0,4
            la $a0,long
            syscall

            C O M P L E T A R

            li $v0,10
            syscall
```

- A partir de la siguiente declaración de un vector de 10 elementos:

```
.data

 array: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
 long: .word 10
 suma :.word 0
```

Haz el código que suma los elementos del vector y calcula el valor medio en coma flotante. Muestra el resultado por la consola.

## Resumen

- Las operaciones de coma flotante en MIPS se hacen en una unidad separada de coma flotante, la FPU, que contiene 32 registros numerados del $f0 al $f31.
- Los números en coma flotante en MIPS se representan según el estándar IEEE754.
- Los valores en doble precisión se representan utilizando parejas de 2 registros, para acceder se tienen que utilizar los registros pares.