

Práctica 9

Coma flotante 2

Objetivos

- Conocer la Unidad de Coma Flotante del MIPS.
- Saber escribir programas de ensamblador utilizando instrucciones de coma flotante del MIPS
- Saber escribir funciones con parámetros en coma flotante y devolver resultados en coma flotante

Material

Simulador MARS y un código fuente de partida

Desarrollo de la práctica. Repertorio de instrucciones, continuación

1. Declaración de valores

Los valores de coma flotante se definen con las directivas `.float` (32 bits) o `.double` (64 bits). Internamente se representarán utilizando el estándar IEEE de coma flotante.

Por ejemplo:

```
pi:    .float 3.1415926536
```

```
otro:  .float -0.325e4 #-0.325 × 104
```

En la tabla 1 se recogen las directivas de declaración de valores en coma flotante.

Declaración	Acción
<code>.float</code>	número de 32 bits en coma flotante IEEE 754
<code>.double</code>	número de 64 bits en coma flotante IEEE 754

Tabla 1. Declaración de tipo en coma flotante

2. Operaciones de lectura y escritura de datos en memoria

Ya hemos visto en la práctica 8 que la primera forma de rellenar los registros de la FPU es utilizando instrucciones de movimiento. La segunda es leyendo o escribiendo una palabra desde o en memoria con las instrucciones:

```
Lwc1 $f1, 40($s1) # "Load word into Coprocessor 1"
Swc1 $f1, 40($s1) # "Store word from Coprocessor 1"
```

Para cargar o guardar un dato en doble precisión habrá que hacer dos operaciones de lectura o escritura para acceder a dos palabras consecutivas de la memoria y obtener un valor de 64 bits.

En la tabla 2 se resumen las instrucciones y pseudoinstrucciones de acceso a la memoria desde la FPU:

Simple precisión	Acción	Doble precisión
lwc1 \$f1, 40(\$s1)	$\$f1 \leftarrow M[\$s1+40]$	ldc1
l.s \$f1, 40(\$s1)	Pseudoinstrucción $\$f1 \leftarrow M[\$s1+40]$	l.d
swc1 \$f1, 40(\$s1)	$M[\$s1+40] \leftarrow \$f1$	Sdc1
s.s \$f1, 40(\$s1)	Pseudoinstrucción $M[\$s1+40] \leftarrow \$f1$	s.d

Tabla 2. Instrucciones y pseudoinstrucciones de acceso a la memoria

- ¿Cuál es la razón por la que el registro base de las instrucciones *lwc1* y *swc1* pertenecen al banco de registros de enteros y no de la FPU?

Código ejemplo 1:

```
# Código ejemplo 3

.data
vector1: .float 5.6e+20, -5.6e+20, 1.2
vector2: .float 1.2, 5.6e+20, -5.6e+20

.text
la $t0, vector1
lwc1 $f0, 0($t0)
lwc1 $f1, 4($t0)
lwc1 $f2, 8($t0)
add.s $f3, $f0, $f1
add.s $f4, $f2, $f3

la $t1, vector2
lwc1 $f5, 0($t1)
lwc1 $f6, 4($t1)
lwc1 $f7, 8($t1)
add.s $f8, $f5, $f6
add.s $f9, $f7, $f8
```

- Haz una traza del programa a mano y obtén el valor de los registros del \$f0 al \$f9. Conviene notar que vector1 y vector2 tienen los mismos elementos, pero con diferente orden. Observa los resultados que hay en \$f4 y \$f9.
- Ensambla, ejecuta el programa y observa el contenido que adquieren los registros para verificar los resultados que has obtenido a mano. ¿Qué conclusión puedes sacar?

3. Instrucciones de comparación

Las instrucciones de comparación permiten hacer varias pruebas entre registros de coma flotante de la FPU, pueden hacer el test de igualdad, menor que y menor o igual. El resultado de la prueba (1 si se cumple, 0 si no se cumple) se almacena en unos registros especiales de un bit. Estos registros son los códigos de condición numerados del 0 al 7. El programador no tiene acceso directo al resultado de las comparaciones, sino que serán leídos por las instrucciones de salto condicional.

Las instrucciones de comparaciones tienen la forma:

```
c.____.s $f2, $f4
```

Aquí lo “____” es un operador de comparación como pueda ser *eq*, *lt* o *le*.

Por ejemplo, la instrucción:

```
c.lt.s $f0,$f1
```

Pone a 1 el código de condición si $\$f0 < \$f1$, en caso contrario lo pone a 0. El valor del código de condición permanecerá con ese valor hasta que otra instrucción le cambie el valor.

En doble precisión también encontramos instrucciones de comparaciones con los mismos operadores. En este caso las instrucciones acaban con *.d*. Por ejemplo:

```
c.eq.d $f0,$f2
```

En la tabla siguiente se recogen las instrucciones de comparación de MIPS en coma flotante:

Simple precisión	Acción	Doble precisión
c.eq.s \$f2, \$f3	si ($\$f2 = \$f3$) flag a 1 si no 0	c.eq.d
c.eq.s 4 \$f0,\$f1	si ($\$f0 = \$f1$) flag 4 a 1 si no 0	c.eq.d 4
c.le.s \$f10, \$f11	si $\$f10 \leq \$f11$) flag a 1 si no a 0	c.le.d
c.le.s 3 \$f10, \$f11	si $\$f10 \leq \$f11$) flag 3 a 1 si no a 0	c.le.d 3
c.lt.s \$f5, \$f8	si $\$f5 < \$f8$ flag a 1 si no a 0	c.lt.d
c.lt.s 2 \$f5, \$f8	si $\$f5 < \$f8$ flag 2 a 1 si no a 00	c.lt.d 2

Tabla 3. Instrucciones de comparación

- No hay comparación *mayor que*, ¿cómo lo podéis solucionar?

Recuerda que hacer el test de igualdad de dos números en coma flotante no es siempre una buena idea porque el resultado de los cálculos en coma flotante no son siempre los mismos. Algunas veces los valores no son iguales, aunque matemáticamente tendrían que serlo. Es mejor utilizar tests del tipo *menor que* o *menor o igual que* en lugar de hacer tests de igualdad.

4. Saltos condicionales para coma flotante

Podemos hacer un salto condicional basado en una comparación en punto flotante en dos pasos. Primero comparamos los dos números flotantes con las instrucciones de comparación y después, basándonos en el resultado de los códigos de condición, se efectúa el salto o no. La instrucción `bc1t` inspecciona el bit de condición y lleva a cabo el salto si es igual a 1. En cambio, la instrucción `bc1f` hace el salto si el bit de condición es 0.

En la tabla 4 se recogen las instrucciones de salto condicional en coma flotante.

Instrucción	Acción
<code>bc1t etiqueta</code>	Salta a etiqueta si el código de condición es 1 (<i>true</i>).
<code>bc1t 2 etiqueta</code>	Salta a etiqueta si el código de condición especificado es 1 (<i>true</i>).
<code>bc1f etiqueta</code>	Salta a etiqueta si el código de condición es 0 (<i>false</i>).
<code>bc1f 4 etiqueta</code>	Salta a etiqueta si el código de condición especificado es 0 (<i>false</i>).

Tabla 4. Instrucciones de salto condicional

- ¿Dónde crees que se ejecutará la instrucción `bc1t`? ¿En la CPU o en la FPU?

Código ejemplo:

```
# Determinar el menor de dos números en coma flotante
...
c.lt.s $f0,$f2    # es < A B?
bc1t  print_A     # sí - escribir en consola A
c.lt.s $f2,$f0    # es B < A?
bc1t  print_B     # sí - Escribir consuela B
...
```

El fragmento de código ejemplo compara dos valores en coma flotante con el objetivo de imprimir en la consola el menor de ellos haciendo un salto a una sección del código que se encarga de hacerlo.

- ¿Qué código de condición se ve afectado? ¿Hasta cuándo permanecerá el valor del código de condición?

Ejercicios a entregar:

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

```
.data
Array: .float 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
long: .word 10
Suma: .float 0
```

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

- Implementar la función `float pow(float x;int n)` que calcula la potencia n-ésima de x. Los argumentos y los valores se pasan según convenio: x en \$f12, n en \$a0. El resultado se devuelve en \$f0. Utilizad el siguiente código de partida:

```
.data
Xpide:.asciiz "X = "
Npide:.asciiz "n = "
powRes: .asciiz "X^n = "
.text

    la $a0, Xpide
    li $v0,4
    syscall
    li $v0,6
    syscall

    la $a0, Npide
    li $v0,4
    syscall
    li $v0,5
    syscall

    mov.s $f12,$f0
    move $a0,$v0
    jal pow

    la $a0,powRes
    li $v0,4
    syscall
    mov.s $f12,$f0
    li $v0,2
    syscall

    li $v0,10
    syscall

pow:    li $t0, 1

        C O M P L E T A R

    jr $ra
```

- Implementar la función *max* que nos devuelve el valor mayor de dos números en coma flotante. Los argumentos se pasan según convenio en \$f12 y \$f14 y el resultado se devuelve en \$f0. Utilizad el siguiente código de partida:

```
.data
Xpide:.asciiz "X = "
Ypide:.asciiz "Y = "
MaxRes: .asciiz "El mayor es "
.text

    la $a0, Xpide
    li $v0,4
    syscall
    li $v0,6
    syscall
    mov.s $f12,$f0

    la $a0, Ypide
    li $v0,4
    syscall
    li $v0,6
    syscall
    mov.s $f14,$f0

    jal max

    la $a0,MaxRes
    li $v0,4
    syscall
    mov.s $f12,$f0
    li $v0,2
    syscall

    li $v0,10
    syscall

max:

    C O M P L E T A R

    jr $ra
```

Resumen

- Las instrucciones de coma flotante de acceso a memoria nos permiten escribir y leer valores en los registros de la FPU.
- Las instrucciones de bot condicional en coma flotante inspeccionan los bits de condición de la FPU que son modificados por las instrucciones de comparación.
- En la página siguiente se recogen todas las instrucciones de coma flotante en una misma tabla.

Instrucciones Aritméticas	
add.s \$f0, \$f1, \$f2	$\$f0 \leftarrow \$f1 + \$f2$
sub.s \$f0, \$f1, \$f2	$\$f0 \leftarrow \$f1 - \$f2$
mul.s \$f0, \$f1, \$f2	$\$f0 \leftarrow \$f1 * \$f2$
div.s \$f0, \$f1, \$f2	$\$f0 \leftarrow \$f1 / \$f2$
sqrt.s \$f0, \$f1	$\$f0 \leftarrow \text{Square root}(\$f2)$
neg.s \$f0, \$f1	$\$f0 \leftarrow \text{Neg}(\$f2)$
abs.s \$f0, \$f1	$\$f0 \leftarrow \text{Abs}(\$f2)$
Instrucciones de Transferencia	
mov.s \$f0, \$f1	$\$f0 \leftarrow \$f1$
mfc1 \$t0, \$f0	$\$t0 \leftarrow \$f0$
mtc1 \$t0, \$f0	$\$f0 \leftarrow \$t0$
lwc1 \$f1, 40(\$s1)	$\$f1 \leftarrow M[\$s1+40]$
l.s \$f1, 40(\$s1)	Pseudoinstrucción $\$f1 \leftarrow M[\$s1+40]$
swc1 \$f1, 40(\$s1)	$M[\$s1+40] \leftarrow \$f1$
s.s \$f1, 40(\$s1)	Pseudoinstrucción $M[\$s1+40] \leftarrow \$f1$
Instrucciones de Conversión de tipo	
cvt.s.w \$f0,\$f2	$\$f0 \leftarrow \text{Conversión a simple precisión del entero en } \$f2$
cvt.w.s \$f0,\$f2	$\$f0 \leftarrow \text{Conversión a entero de } \$f2$
ceil.w.s \$f0,\$f2	$\$f0 \leftarrow \text{Asigna menor número entero igual o mayor que } \$f2$
floor.w.s \$f0,\$f2	$\$f0 \leftarrow \text{Asigna mayor número entero igual o menor que } \$f2$
trunc.w.s \$f0,\$f2	$\$f0 \leftarrow \text{Asigna el entero truncado de } \$f2$
Instrucciones de comparación y salto condicional	
c.eq.s \$f2, \$f3	si ($\$f2 = \$f3$) flag a 1 si no 0
c.eq.s 4 \$f0,\$f1	si ($\$f0 = \$f1$) flag 4 a 1 si no 0
c.le.s \$f10, \$f11	si $\$f10 \leq \$f11$) flag a 1 si no a 0
c.le.s 3 \$f10, \$f11	si $\$f10 \leq \$f11$) flag 3 a 1 si no a 0
c.lt.s \$f5, \$f8	si $\$f5 < \$f8$ flag a 1 si no a 0
c.lt.s 2 \$f5, \$f8	si $\$f5 < \$f8$ flag 2 a 1 si no a 00
bc1t etiqueta	Salta a etiqueta si el código de condición es 1 (<i>true</i>).
bc1t 2 etiqueta	Salta a etiqueta si código de condición especificado es 1 (<i>true</i>).
bc1f etiqueta	Salta a etiqueta si el código de condición es 0 (<i>false</i>).
bc1f 4 etiqueta	Salta a etiqueta si código de condición especificado es 0 (<i>false</i>).

Tabla 5. Instrucciones de coma flotante