Pràctica 8
Coma flotant 1

8.1 Objectius

- Recordar i afermar la representació de nombres en coma flotant segons l'estàndard IEEE 754
- Conèixer la Unitat de Coma Flotant del MIPS.
- Saber escriure programes d'assemblador utilitzant instruccions de coma flotant del MIPS

8.2 Material

Simulador MARS i un codi font de partida

8.3 Introducció teòrica

8.3.1 Representació de la coma flotant

En la majoria dels computadores, els nombres reals es representen de foma binària utilitzant el format estandarditzat IEEE 754 (Institute of Electrical and Electronic Engineers Standard 754). Hi ha tres formats bàsics de grandària diferent: simple precisió (32 bits), doble precisió (64 bits) i doble precisió estesa (128 bits). Els bits estan dividits en tres camps de grandària fixa: el primer (S) és el signe de grandària 1 bit, el segon és l'exponent (E) i el tercer és la mantissa (M). La diferència entre els tres formats radica en el nombre de bits dels camps exponent i mantissa. Així, en simple precisió l'exponent és de 8 bits, en doble precisió d'11 bits i en doble precisió estesa de 15 bits. La mantissa té 23 bits en simple precisió, 52 en doble precisió i 112 en doble precisió estesa.

El bit de signe, S, indica el signe del nombre, 0 positiu i 1 negatiu. L'exponent E és positiu i segueix una notació esbiaixada per a incloure el signe, està representat en excés 2^{q-1}-1, on q són els nombres de bits de l'exponent. En simple precisió l'excés és 127 (q=8).

La mantissa està normalitzada amb un 1 implícit a l'esquerra de la coma decimal. Per tant, els valors representats de la mantissa estan compresos entre 1,0000... i 1,1111...

El valor 0 i el valor 2^q-1 (tot a uns) per a l'exponent estan reservats per als casos especials que es mostren en la taula 1:

Exponent	Mantissa	Valor
2^q-1	≠0	NaN (Not a Number)
2^q-1	0	+∞ i -∞ segons el signe S
0	0	+0 i -0 segons el signe S
0	≠0	Nombres desnormalitzats ($0.M \times 2^{-126}$)

Taula 1: Casos especials del Format IEEE 754.

La representació del format d'un valor en coma flotant en simple precisió de 32 bits es mostra en la figura 1:

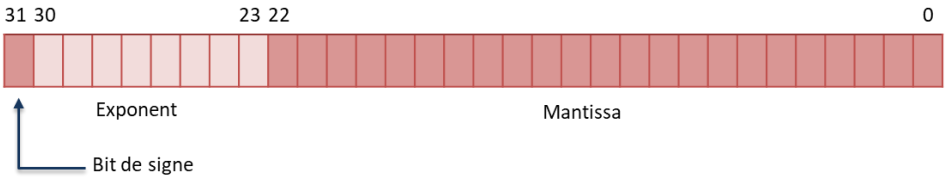

Figura 1: Format IEEE 754 en simple precisió.

El valor representat d'un nombre en coma flotant en simple precisió és:

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

8.3.2 La coma flotant en MIPS

El processador MIPS opera amb els nombres enters en la unitat central de processament (CPU). En canvi, les operacions de coma flotant es fan en una unitat separada de coma flotant, l'FPU (Floating Point Unit) que en MIPS rep el nom de *coprocessador* 1. Aquest coprocesador opera amb nombres representats en coma flotant en simple precisió (32 bits) i doble precisió (64 bits).

L'FPU del MIPS té un conjunt especial de 32 registres de 32 bits numerats del $f0 al $f31. Cadascun pot emmagatzemat nombres en coma flotant de simple precisió. Per a emmagatzemar nombres de doble

precisió de 64 bits es necessiten parelles d'aquests registres. Això significa que sols es podran utilitzar registres de coma flotant amb numeració parella per a referir-se a ells. Per exemple, quan s'emmagatzema un valor de doble precisió en el registre $f0 realment està guardant-se en la parella de registres $f0 i $f1.

Cada una de les unitats (CPU i FPU) té les seues pròpies instruccions aritmètiques per a operar amb cada tipus de dades i les seus pròpies instruccions d'accés a la memòria. A més, existeixen una sèrie d'instruccions de transferència per a moure dades entre els registres de les dues unitats. En la figura 2 es mostra el model del processador MIPS.

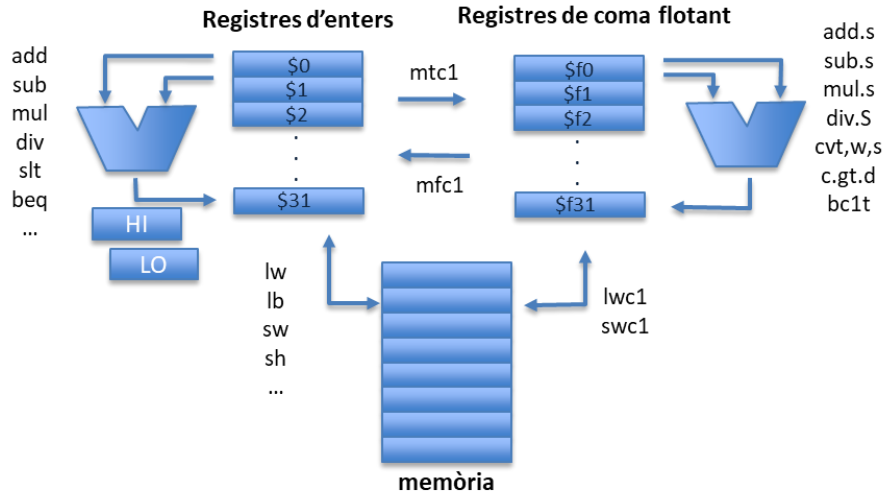

Figura 2. Model del processador MIPS.

A més a més, el coprocessador de coma flotant té vuit indicadors de codi de condició (cc) numerats del 0 al 7. Les instruccions de comparació els utilitzen per a guardar els resultat de les comparacions i les instruccions de bot i moviment condicional per a decidir si fan l'operació o no.

Per qüestions de simplicitat utilitzarem sols nombres en coma flotant representats en simple precisió (32 bits), encara que indicarem també les instruccions en doble precisió.

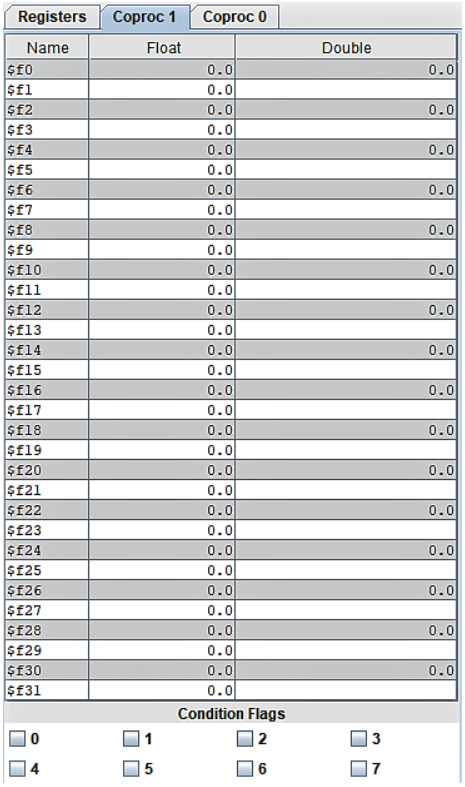

Figura 3. Registres mostrats per MARS junt amb els codis de condició.

Aquests registres junts amb els codis de condició es poden veure en el MARS seleccionant la solapa *Coproc 1* com mostra la figura 3:

També en el cas dels registres en coma flotant (de manera similar als registres de la CPU) s'utilitza un conveni d'ús. No seguir-lo podria portar problemes en el pas de paràmetres, obtenció de resultats de funcions i utilització dels registres a través de les crides a subrutines. En la taula 2 es mostra el conveni d'utilització dels registres de coma flotant del MIPS.

Registre	Utilització
$f0..$f3	Utilitzat per a contenir els resultat de les funcions de tipus coma flotant
$f4..$f11	Registres temporals utilitzats per a avaluar expressions , els seus valors NO estan preservats per les crides a subrutines.
$f12 .. $f15	Utilitzats per a passar paràmetres a subrutines , els seus valors NO estan preservats a través de les crides a subrutines
$f16..$f19	Més registres temporals. NO preservats a través de crides a subrutines
$f20..$f30	Registres de guarda, els seus valors estan preservats a través de les crides a subrutines.

Taula 2. Conveni d'utilització dels registres en coma flotant del MIPS.

El simulador MARS té una eina de representació de la coma flotant que il·lustra els nombres en coma flotant de simple precisió. Vegeu `Tools-> Floating Point Representation` i obriu la finestra. Vegeu el que il·lustra la figura 4

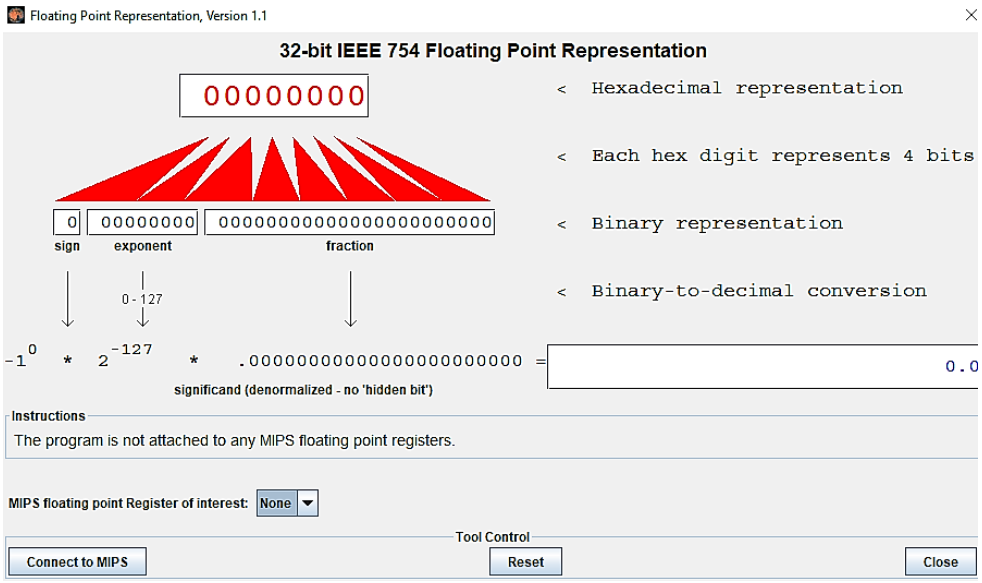

Figura 4. Eina de representació en coma flotant del MARS.

Es pot fer ús d'aquesta eina per a comprovar el valor decimal i binari dels nombres en coma flotant.

8.3.2.1 Activitat 1

- Utilitzeu l'eina de representació en coma flotant del MARS i comproveu que el nombre decimal de C0E00000 és -7.0 i que la representació en 32 bits de simple precisió de 4.25 és 40880000

8.4 Desenvolupament pràctic. Joc d'instruccions

8.4.1 Operacions aritmètiques

Com que l'FPU és una unitat separada, les operacions en coma flotant sols usen els registres de la unitat. Les instruccions MIPS per a sumar i restar nombres en coma flotant de simple precisió són:

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

En l'últim exemple, el registre $f0 contindrà el resultat de la resta. A diferència del registre $0 del banc de registres de la CPU, l'FPU no té cap registre amb valor 0.

De manera similar, excepte perquè no es poden utilitzar registres amb numeració imparella, podem operar amb els nombres en coma flotant de doble precisió amb les instruccions següents:

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

Fixeu-vos que sols s'indica en les instruccions de doble precisió el registre que conté els 32 bits de més pes del nombre que és el que té numeració parella.

S'indica que és una instrucció de simple precisió amb l'extensió *.s* i una de doble precisió amb l'extensió *.d*.

Les instruccions següents són sintàcticament incorrectes, no estan permeses:

```
add.s $s0, $s0, $s1  # No està permesa, ja que add.s espera
                     # registres FPU
add   $f0, $f2, $f2  #No permesa perquè add espera registres CPU

add.d $f0, $f2, $f5  #No està permesa perquè f5 és imparell
addi.s $f0, $f1, 2.3 #No hi ha instruccions immediates
                     #en coma flotant
```

Per a la multiplicació i la divisió es tenen les instruccions següents en simple precisió (extensió *.s*) i doble precisió (extensió *.d*):

```
mul.s $f0, $f1, $f2 #Multiplicació en simple precisió
div.s $f0, $f1, $f2 #Divisió en simple precisió

mul.d $f0, $f0, $f2 #Multiplicació en doble precisió,
div.d $f0, $f0, $f2 #Divisió en doble precisió
```

Qüestió 1

- Quina és la raó per la qual no hi han instruccions aritmètiques en coma flotant amb dades immediates?

Qüestió 2

- Per què no hi ha registres *HI* i *LO* per a guardar el resultat de la multiplicació i divisió en coma flotant de la mateixa manera que amb els nombres enters?

Totes les instruccions aritmètiques de coma flotant segueixen un format d'instrucció tipus R amb el mateix codi d'operació: 0x11. En la taula 3 resumim les instruccions aritmètiques.

Simple precisió	Acció	Doble precisió
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
mul.s $f0, $f1, $f2	$f0←$f1*$f2	mul.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

Taula 3. Instruccions aritmètiques en coma flotant.

8.4.2 Operacions de moviment de dades

Per a poder fer qualsevol tipus d'operació aritmètica s'han d'omplir els registres FPU i després de fer-se les operacions el resultat s'ha de guardar en un altre registre FPU. Hi ha dues maneres de moure dades des de o cap als registres de coma flotant, mitjançant registres o bé mitjançant la memòria.

La primera forma és moure paraules des de o cap als registres CPU. Aquesta operació la realitzen les instruccions:

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

Fixeu-vos amb l'ordre dels operands d'aquestes instruccions.

El MIPS també permet moure dades entre els mateixos registres de l'FPU amb la instrucció `move`:

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

En la taula 4 es recullen les instruccions de moviment de dades entre registres.

Simple precisió	Acció	Doble precisió
mov.s $f0, $f1	$f0 ← $f1	mov.d
mfc1 $t0, $f0	$t0 ← $f0	
mtc1 $t0, $f0	$f0 ← $t0	

Taula 4. Instruccions moviment de dades en coma flotant.

Aprofundim a continuació en les instruccions en coma flotant a partir del codi exemple següent:

```
        ##################################
        #                                #
        # Codi exemple de l'activitat 2  #
        #      De la CPU a l'FPU         #
        #                                #
        ##################################


li $t0, 0xFF800000   # Menys infinit
mtc1 $t0, $f12       # movem $f0 -> $f12=0xFF800000

li $t1, 0x7F8003A0   # Not a Number (NaN)
mtc1 $t1, $f20       # movem $f0 -> $f20=0x7F8003A0
```

El codi fica un valor en hexadecimal en $t0 i en $t1. Segons l'estàndard IEEE 754, el valor en $t0 representa el -∞, i el que hi ha en $t1 representa un NaN. El programa passa aquests valors als registres $f12 i $f20 de l'FPU.

8.4.2.1 Activitat 2

- Analitzeu el *codi exemple de l'activitat 2*. Fent ús de l'eina de representació en coma flotant del MARS, comproveu que realment en $f12 hi ha un -∞, i en $f20, un *NaN* segons l'estàndard IEEE 754.

- Feu que el contingut de $f1 siga el valor 1 en coma flotant i el de $f2 el valor -2.5 en coma flotant.

Qüestió 3

- Digueu una manera d'escriure un 0.0 en $f0 amb només una instrucció de màquina.

8.4.3 Conversió de tipus

Si ens interessa fer una operació amb dos operands font (suma, resta, multiplicació...), els dos han de ser del mateix tipus. Si un d'ells és un valor enter i l'altre un valor en coma flotant, aleshores un dels operands s'ha de convertir al tipus de l'altre, ja que l'operació s'haurà de fer amb els circuits de coma flotant o bé amb els circuits d'enters. En qualsevol cas, la conversió es farà en l'FPU, tant si és de flotant a enter com si és d'enter a flotant.

La instrucció MIPS que fa la conversió es `cvt._._` en què el subratllat s'ompli amb el símbol *s* (per a coma flotant de simple precisió), amb *d* (per a coma flotant de doble precisió), o amb *w* (per a un enter). Per exemple:

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

agafa els 32 bits que hi ha en $f0 i que representen un valor enter i els converteix en un nombre en coma flotant del mateix valor. El valor resultant el guarda en $f0.

En la taula 5 es recullen les instruccions de conversió de tipus del MIPS.

Simple precisió	Acció	Doble precisió
cvt.s.w $f0,$f2	$f0←Conversió a simple precisió de l'enter en $f2	cvt.s.d
		cvt.d.w
		cvt.d.s
cvt.w.s $f0,$f2	$f0←Conversió a enter de $f2	cvt.w.d
ceil.w.s $f0,$f2	$f0 ←Assigna el menor nombre enter igual o major que $f2	ceil.w.d
floor.w.s $f0,$f2	$f0 ←Assigna el major nombre enter igual o menor que $f2	floor.w.d
trunc.w.s $f0,$f2	$f0 ←Assigna l'enter truncat de $f2	trunc.w.d

Taula 5. Instruccions de conversió de tipus en coma flotant

Aprofundim a continuació en les instruccions de conversió de tipus a partir del fragment de codi exemple següent:

```
        ###################################
        #                                 #
        # Codi exemple de l'activitat 3   #
        #      Conversió de tipus         #
        #                                 #
        ###################################

              .....

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

El codi fica el valor -8 en $s0 i el passa al registre $f0 de l'FPU; si no convertim la dada que hi hauria en $f0 no correspondria al valor -8.

8.4.3.1 Activitat 3

- Analitzeu el *codi exemple de l'activitat 3*. Fent ús de l'eina de representació en coma flotant del MARS, comproveu que realment en $f0, després de la conversió, hi ha un -8.

- Quin valor consideraria la màquina que hi hauria en $f0 si no férem la conversió amb la instrucció `cvt.s.w`? Aprofiteu que podeu veure continguts en decimal i en hexadecimal.

Qüestió 4

- Feu que el contingut de $f1 siga el valor 1 en coma flotant i el de $f2 el valor -2 en coma flotant utilitzant les instruccions de conversió de tipus.

8.4.3.2 Activitat 4

Donat el següent fragment de codi exemple:

```
        ###################################
        #                                 #
        # Codi exemple de l'activitat 4   #
        #                                 #
        ###################################

              .....


li $s0, 841242345
mtc1 $s0, $f0     # movem de $s0 a $f0
cvt.s.w $f1, $f0  # Conversió d'enter a simple precisió
```

```
cvt.w.s $f1, $f1  # Conversió de simple precisió a enter
mfc1 $s1, $f1     # movem de $f1 a $s1
```

- Assembleu el codi i comproveu el resultat.
- Quina és la raó per la qual en finalitzar el programa els continguts de $s0 i $s1 són diferents?

8.4.3.3 Activitat 5

Considereu el següent codi exemple:

```
        ##################################
        #                                #
        # Codi exemple de l'activitat 5  #
        #                                #
        ##################################

             .....

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
```

- Assembleu i executeu el codi. Quin valor representa en el format IEEE 754 el contingut final de $f0?
- Observeu que no ha ocorregut cap excepció quan s'executa el codi. Ocorre el mateix si dividim per zero? I si fem 0/0? Afegiu instruccions al codi anterior i feu la prova.

8.4.3.4 Activitat 6

Considereu el següent codi exemple:

```
        ##################################
        #                                #
        # Codi exemple de l'activitat 6  #
        # detectar casos especials del   #
        # format 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 comprovar en $f0

        mfc1 $s0,$f0
        lw $t4,mmask # carregar màscara de la mantissa
        and $t0,$s0,$t4 # extraure mantissa de $s0
        lw $t4,emask # carregar màscara de l'exponent
        and $t2,$s0,$t4 # extraure exponent de $s0
        srl $t2,$t2,23 # desplaçar exponent

        lw $t3,exp1 #carreguem valor exponent tot a uns
        beq $t2,$t3,exp_a_1 #exponent tot a uns?
```

S'ha afegit al codi de l'activitat 5 un fragment de codi que extrau l'exponent i la mantissa d'un nombre representat en format IEEE 754 per a detectar si s'ha produït desbordament, i en aquest cas mostrar un missatge d'avís en la consola. El codi està incomplet.

- Analitzeu el codi i observeu com s'extrau l'exponent i la mantissa.

Qüestió 5

- Completeu el codi de l'activitat 6 perquè mostre en consola un missatge d'error per desbordament. Comproveu que funciona correctament.

Qüestió 6

- Completeu el codi de l'activitat 6 perquè detecte tots els casos especials i mostre missatges en la consola. Feu diferents proves i comproveu que funciona.

8.4.3.5 Crides al sistema

Es pot llegir un nombre en coma flotant de teclat i escriure'l en consola utilitzant la instrucció *syscall* passant el nombre del servei en $v0 com mostra la taula 8:

Servei	Codi de crida	Arguments	Resultat

Taula 8. Codis de servici de syscall per a coma flotant.

Print Float	2	$f12=Flotant en simple precisió	Imprimir en consola el contingut de $f12
Print Double	3	$f12=Flotant en doble precisió	Imprimir en consola el contingut de $f12
Read Float	6		Llig de la consola un flotant en simple precisió i el guarda en $f0
Read Double	7		Llig de la consola un flotant en simple precisió i el guarda en $f0

Qüestió 7

- Completeu el següent codi de partida que demana el radi per teclat i ha de calcular i mostrar en la consola la longitud de la circumferència i l'àrea del cercle.

```
        ###################################
        #                                 #
        # Codi de partida de la qüestió 7 #
        # Càlcul longitud circumferència  #
        #     Càlcul àrea del cercle      #
        #                                 #
        ###################################

        .data
demanaPi:    .asciiz "Dóna'm el valor de pi..."
demanaRadi:  .asciiz "Dóna'm el radi... "
long:  .asciiz "Longitud de la circunferència = "
super: .asciiz "Àrea del cercle = "

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

Qüestió 8

A partir de la següent declaració d'un vector de 10 elements:

```
        ###################################
        #                                 #
```

```
          # Codi de partida de la qüestió 8   #
          #                                   #
          #####################################

.data

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

- Feu el codi que suma els elements del vector i calculeu el valor mitjà en coma flotant. Mostreu el resultat per la consola.

8.5 Resum

- Les operacions de coma flotant en MIPS es fan en una unitat separada de coma flotant, l'FPU, que conté 32 registres numerats del $f0 al $f31.
- Els nombres en coma flotant en MIPS es representen segons l'estàndard IEEE754.
- Els valors en doble precisió es representen utilitzant parelles de 2 registres; per accedir-hi s'han de numerar els registres parells.