

# Estructura de los computadores

Practica 7 8 9

Francisco Joaquín Murcia Gómez 48734281H

Grupo 3

# Índice

**Practica 4.....3-10**

Aritmética de enteros (3) y funciones

Ejemplos .....3-8

Entregas .....9-10

**Practica 5.....11-14**

Estructuras de control

Ejemplos .....11-12

Entregas .....13-14

**Practica 6.....15-18**

Variables

Ejemplos .....15

Entregas .....16-18

# Practica 7

## Ejemplos

### 1. Vectores de caracteres

#### 1.1. ¿Cuántos caracteres tiene la cadena?

30

#### 1.2. ¿En qué dirección de la memoria se encuentra el carácter null)?

1e (1c+4)

#### 1.3. ¿Cómo se actualiza el índice del vector?

\$S1

#### 1.4. ¿El programa funcionaría si la cadena solo constara del carácter null?

Si, pero terminaría inmediatamente

#### 1.5. Modifica el código para que muestre por pantalla el mensaje "El número de caracteres de la cadena es: " y a continuación el resultado.

```
#Contar caracteres de una cadena
.data
str:.ascii "Estructuras de los"
.asciiz "Computadores"
A:.asciiz "el numero de cadenas es : "
.text
la $s0, str
add $s1, $zero, $zero
# Iniciamos contador a 0
loop:
add $t0, $s0, $s1 # dirección del byte a examinar
lb $t1, 0( $t0 )
beq $t1, $zero, exit # salimos si carácter leído='\0'
addi $s1, $s1, 1 # incrementamos el contador
j loop

exit:
la $a0, A
li $v0, 4
syscall

move $a0, $s1
li $v0, 1
syscall

li $v0, 10
syscall
```

ssages

Run I/O

```
el numero de cadenas es : 30
-- program is finished running --
```

#### 1.6. Modifica el código para que calcule el número de veces que se repite la vocal "u"

```
el numero de u es: 3
el numero de cadenas es : 30
-- program is finished running --
```

```

.asciiz "Computadores"
A:.asciiz "el numero de cadenas es : "
B:.asciiz "el numero de u es: "
.text
la $s0, str
add $s1, $zero, $zero
li $s2, 0 #contador de u a 0
li $s3, 'u' #u a ascii e un registro
# Iniciamos contador a 0
loop:
add $t0, $s0, $s1 # dirección del byte a examinar
lb $t1, 0( $t0 )

bne $t1,$s3,no_u #si es u continua, si no salta y no suma
addi $s2,$s2,1
no_u:
beq $t1, $zero, exit # salimos si carácter leído='\0'
addi $s1, $s1, 1 # incrementamos el contador
j loop
exit:
#cadena b
la $a0,B
li $v0, 4
syscall
move $a0,$s2
li $v0, 1
syscall
li $a0,'\n'
li $v0, 11
syscall
#cadena b
la $a0,A
li $v0, 4
syscall
move $a0,$s1
li $v0, 1
syscall
li $v0, 10
syscall

```

## 2. Vectores de enteros

**2.1. Ensambla y ejecuta el programa. Comprueba que el vector A se copia en el vector B. ¿En qué dirección empieza el vector B?**

0x10010014

**2.2. ¿Porque no se pueden acabar los vectores con el carácter null igual que se hace con las cadenas de caracteres?**

Porque no son caracteres, son enteros

**2.3. En el programa se recorren los vectores actualizando el índice con la instrucción sll. ¿De qué otra manera se podrían recorrer los vectores?**

Con un acumulador de 4

**2.4. Se ha utilizado un bucle del tipo do-while, modifica el programa por que el bucle sea de tipo for-while.**

```

#Ejemplo: recorrer un vector de enteros

.data
A: .word 2, 4, 6, 8, 10 # vector A iniciado con valores
B: .word 0:5 # Vector B vacío
C: .space 50 # Otra definición de vector vacío

.text
la $s0, A # Dirección base del vector
la $s1, B # Dirección base del vector B
li $s5, 5 # Tamaño del vector

li $v0, 5

loop: beq $s2, $s5, exit

add $t1, $s0, $t0
add $t2, $s1, $t0

addi $s2, $s2, 1
lw $t3, 0($t1) # Copia el vector A en el vector B.
sw $t3, 0($t2)
sll $t0, $s2, 2

j loop

exit: li $v0, 10
      syscall

```

## 2.5. Modifica el programa para que el vector B se rellene con enteros leídos del teclado. Previamente se tiene que mostrar un mensaje por consola que pida los elementos

```

.data
A: .word 2, 4, 6, 8, 10 #vector A iniciado con valores
B: .word 0:5 # Vector B vacío
C: .space 50 #Otra definición de vector vacío
texto: .asciiz "Introduce numero: "

.text
la $s0, A
la $s1, B
li $s5, 5

loop: add $t1, $s0, $t0
      add $t2, $s1, $t0
      addi $s2, $s2, 1 #Índice del vector.

      la $a0, texto
      li $v0, 4
      syscall

      move $a0, $s1
      li $v0, 5
      syscall
      move $t3, $v0

      li $a0, '\n'
      li $v0, 11
      syscall

      sw $t3, 0($t2)
      sll $t0, $s2, 2 #Índice del vector x4.
      bne $s2, $s5, loop

      li $v0, 10
      syscall

```

```
Reset: reset completed.
```

```
Introduce numero: 5
```

```
Introduce numero: 6
```

```
Introduce numero: 3
```

```
Introduce numero: 4
```

```
Introduce numero: 8
```

### **3. Direccionamiento en memoria**

#### **3.1. ¿Cuántas pseudoinstrucciones contiene el código?**

2, "la" y "sw"

#### **3.2. Ensambla el código y observa la traducción de las pseudoinstrucciones en instrucciones del MIPS. ¿En qué instrucciones se ha traducido sw \$t3, C?**

¿Qué registro auxiliar se ha utilizado?

En "lui" y "sw", \$1

# Entregas

Haz el código de la función `sum` que calcula la suma de los valores positivos y negativos del vector, dirección del cual se pasa como parámetro en `$a0` y la longitud en `$a1`. La función devuelve en `$v0` la suma de los valores positivos y en `$v1` la suma de los negativos. Recuerda que en la función tienes que utilizar los registros `$tj`.

```

.data
vector: .word -4, 5, 8, -1
msg1: .asciiz "\n La suma de los valores positivos és = "
msg2: .asciiz "\n La suma de los valores negativos és = "

.text

Principal:

li $v0, 4 # Función para imprimir string
la $a0, msg1 # Leer la dirección de msg1
syscall
la $a0, vector # dirección del vector como parámetro
li $a1, 4 # Longitud del vector como parámetro
# inicializo contador y sumas
li $v0, 0
li $v1, 0
li $t0, 0 # contador

jal sum # Llamada a la función sum

move $a0, $v0 # Resultado 1 de la función
li $v0, 1
syscall # Imprimir suma positivos
li $v0, 4
la $a0, msg2

syscall
li $v0, 1
move $a0, $v1 # Resultado 2 de la función
syscall # imprimir suma negativos

li $v0, 10 # Acabar programa
syscall

```

Funciones:

```

#-----funciones-----#
sum:
    blt $t0,$a1 bucle# si es mayor a 4 salta
    jr $ra
    bucle:

    lw $t3,0($a0)
    addi $a0,$a0,4 # paso a la siguiente posicion de memoria
    addi $t0,$t0,1# aumento del contador

    bltz $t3,negativo#si numero en posicion es negativ es negativo salta

    add $v0,$v0,$t3# suma positivos
    j sum

    negativo:
    add $v1,$v1,$t3# suma negativos
    j sum

```

Consola:

```

La suma de los valores positivos és = 13
La suma de los valores negativos és = -5
-- program is finished running --

```

Ventana de registros \$v0,\$v1

\$v0	2	13
\$v1	3	-5

**Haz el código que calcula la suma de los elementos de la diagonal principal de una matriz 4x4 de valores enteros introducida por teclado. Muestra la suma por pantalla.**

main



```

.data
    matriz: .byte 0:15
    men1: .asciiz "escribe los numeros de la matriz de la posicion 0 a la 15: \n"
    men2: .asciiz "\n la suma de la diagonal es: "

.text
li $a1,16 # longitud de la matriz
li $t0,0 # contador
la $a2,matriz # direccion de la matriz
la $t1, matriz # direccion de la matriz

li $v0, 4
la $a0, men1
syscall

jal leer # pedir y guardar numeros

jal sum # sumar la diagonal

li $v0, 4
la $a0, men2
syscall

li $v0,1
move $a0,$a3
syscall

li $v0, 10 # Acabar programa
syscall

```

### Funciones:

```

#-----funciones-----#
leer:
    blt $t0,$a1 bucle# si es menor a 16 salta
    jr $ra
    bucle:
        li $v0 5
        syscall

        sb $v0,0($t1)# se cargan los datos de la matriz

        addi $t1,$t1,1# aumento de la direccion de memoria
        addi $t0,$t0,1# aumento del contador
        j leer

sum:
    lb $t3,0($a2)#posicion 0
    add $a3,$a3,$t3# sumo lo que llevo + posicion

    lb $t3,5($a2)#posicion 5
    add $a3,$a3,$t3# sumo lo que llevo + posicion

    lb $t3,10($a2)#posicion 10
    add $a3,$a3,$t3# sumo lo que llevo + posicion

    lb $t3,15($a2)#posicion 15
    add $a3,$a3,$t3# sumo lo que llevo + posicion

    jr $ra

```

### Funcionamiento:

Introducimos la matriz  $A=[1,2,3,4; 1,2,3,4; 1,2,3,4; 1,2,3,4]$

```
escribe los numeros de la matriz de la posicion 0 a la 15:
```

```
1  
2  
3  
4  
1  
2  
3  
4  
1  
2  
3  
4  
1  
2  
3  
4
```

```
la suma de la diagonal es: 10
```

Introducimos la matriz  $B=[1,1,1,1; 1,1,1,1; 1,1,1,1; 1,1,1,1]$

```
escribe los numeros de la matriz de la posicion 0 a la 15:
```

```
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1
```

```
la suma de la diagonal es: 4
```

# Practica 8

## Ejemplos

### 1. Operaciones aritméticas

#### 1.1. ¿Cuál es la razón por la que no hay instrucciones aritméticas en coma flotante con datos inmediatos?

Porque hay que pasarlo a IEEE

#### 1.2. ¿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?

Porque te lo da en IEEE

### 2. Operaciones de movimiento de datos

#### 2.1. Haciendo uso de la herramienta de representación en coma flotante del MARS, comprueba que realmente en \$f12 hay un $-\infty$ y en \$f20 un NaN según el estándar IEEE 754.

**32-bit IEEE 754 Floating Point Representation**

Hexadecimal representation: **FF800000**

Each hex digit represents 4 bits

Binary representation: **11111111 000000000000000000000000**

Binary-to-decimal conversion:

Exponent: 11111111 (255) - 127 = 128

Fraction: 000000000000000000000000

Significand ("hidden bit" underlined):  $1.000000000000000000000000$

Result: **-Infinity**

---

**32-bit IEEE 754 Floating Point Representation**

Hexadecimal representation: **7F8003A0**

Each hex digit represents 4 bits

Binary representation: **01111111 000000000000011101000000**

Binary-to-decimal conversion:

Exponent: 01111111 (127) - 127 = 0

Fraction: 000000000000011101000000

Significand ("hidden bit" underlined):  $1.000000000000011101000000$

Result: **NaN**

#### 2.2. Haz que el contenido de \$f1 sea el valor 1 en coma flotante y el de \$f2 el valor -2.5 en coma flotante.

\$f0	0.0
\$f1	1.0
\$f2	2.5
\$f3	0.0

```

1  li $t0, 0x3F800000
2  mtc1 $t0, $f1 # movemos $t0 -> $f1=0x3F800000
3  li $t1, 0x40200000
4  mtc1 $t1, $f2 # movemos $t1 -> $f2=0x40200000

```

2.3. Di una manera de escribir un 0.0 en \$f0 con sólo una instrucción máquina.

`mtc1 $zero,$f0`

### 3. Conversión de tipo

3.1. Haciendo uso de la herramienta de representación en coma flotante del MARS, comprueba que realmente en \$f0 después de la conversión hay un -8.

3.2. ¿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.

Un NaN

3.3. 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.

```

mpst.asm
1  li $s0, 1
2  mtc1 $s0, $f0 # movemos $s0 -> $f0
3  cvt.s.w $f0, $f0

li $s1, 2
mtc1 $s1, $f2 # movemos $s1 -> $f2
cvt.s.w $f2, $f2

```

3.5. ¿Cuál es la razón por la que al finalizar el programa los contenidos de \$s0 y \$s1 son distintos?

Por el redondeo

3.6. Ensambla y ejecuta el código. ¿Qué valor representa en el formato IEEE 754 el contenido final de \$f0?

Infinito

3.7. 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.

\$f0	Infinity
\$f1	NaN

```

li $s0, 1

mtcl $s0, $f0
cvt.s.w $f0, $f0

mtcl $zero, $f1
cvt.s.w $f1, $f1

div.s $f0, $f0, $f1 # 1/0 -> $f0
div.s $f1, $f1, $f1 # 0/0 -> $f1

```

**3.8. Completa el código para que muestre en consola un mensaje de error por desbordamiento. Comprueba que funciona correctamente**

```

.text
addi $s0, $0, 1
sll $s0, $s0, 30 # $s0 = 2^30
mtcl $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

mfcl $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?
j correcto
exp_a_1:
    la $a0, A
    li $v0, 4
    syscall
    j exit

correcto:
    la $a0, B
    li $v0, 4
    syscall
    j exit

exit:
    li $v0, 10
    syscall

hay desbordamiento
-- program is finished running --

```

**3.9. Completa el código para que detecte todos los casos especiales y muestre mensajes en la consola. Haz distintas pruebas y comprueba que funciona.**

```

# detectar casos especiales del formato IEEE 754
.data
mmask: .word 0x007FFFFF
emask: .word 0x7F800000 #todo unos
expl: .word 255
A:.asciiz "infinito"
B:.asciiz "cero"
C:.asciiz "NaN"
D:.asciiz "valores normales"
.text
addi $s0, $0, 1
sll $s0, $s0, 30 # $s0 = 2^30
mtcl $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

mfcl $s0,$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,expl #cargamos valor exponente todo a unos
beq $t2,$t3,todo1 #exponente todo a unos?
j correcto

todo1:
    beq $t0,$s0,mant_no_0
    la $a0, A
    li $v0, 4
    syscall
    j exit
mant_no_0:
    la $a0, C
    li $v0, 4
    syscall
    j exit
correcto:
    beq $t0,$s0,mant_no_0_2
    la $a0, B
    li $v0, 4
    syscall
    j exit
mant_no_0_2:
    la $a0, D
    li $v0, 4
    syscall
    j exit
exit:
    li $v0,10
    syscall

```

# Entregas

**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.**

El main del programa

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

li $v0,4
la $a0,demanaPi
syscall

li $v0,6 # pido flotante
syscall
mov.s $f1, $f0 # pi --> $f1

li $v0,4
la $a0,pideRadio
syscall

li $v0,6 # pido flotante
syscall
mov.s $f2, $f0 # radio --> $f2

li $v0,4
la $a0,long
syscall

jal longitud # 2*pi*radio

li $v0,4
la $a0,super
syscall

jal area # pi*radio^2

li $v0,10
syscall
```

Funciones del programa:

```

#-----funciones-----#
longitud:

    li $a1, 0x40000000 #2 en IEEE
    mtcl $a1, $f4 # metemos 2 en $f4

    mul.s $f3,$f1,$f2 # pi*radio
    mul.s $f5,$f3,$f4 #pi*radio*2

    #mostramos en pantalla
    mov.s $f12, $f5
    li $v0,2
    syscall

    # salto de linea
    li $a0,'\n'
    li $v0,11
    syscall

    jr $ra

area:

    mul.s $f3,$f2,$f2 # radio^2
    mul.s $f4,$f3,$f1 # radio^2*pi

    #mostramos en pantalla
    mov.s $f12, $f4
    li $v0,2
    syscall

    jr $ra

```

## Pruebas:

Dame el valor de pi...3.14	Dame el valor de pi...3.141592653589793238
Dame el radio... 5	Dame el radio... 4
Longitud de la circunferència = 31.400002	Longitud de la circunferència = 25.132742
Área del círculo = 78.5	Área del círculo = 50.265484
-- program is finished running --	

El radio puede ser tambien en coma flotante

```

Dame el valor de pi...3.141892
Dame el radio... 5.45875847
Longitud de la circunferència = 34.30166
Área del círculo = 93.62223

```

Si el radio es muy grande el resultado te lo pone en notacion cientifica

```

Dame el valor de pi...3.14159
Dame el radio... 5000000000000
Longitud de la circunferència = 3.14159011E13
Área del círculo = 7.853975E25

```



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

Haz el código que suma los elementos del vector y calcula el valor medio.

Muestra el resultado por la consola.

```
.data
A: .asciiz "la suma de los elementos del vector es : "
B: .asciiz "la media es: "

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

la $t0, array
lw $a1, long #cargamos el tamaño
li $t1,0 # contador
# suma de los diferentes elementos del array
inicio:
    beq $a1,$t1,final# salta si el contador es 10

    lw $t2,0($t0) #carga en $t2 el elemento de la direccion de $t0
    mtcl $t2,$f0 # guardamos $t2->$f0
    cvt.s.w $f0,$f0# $f0-->IEEE

    add.s $f1,$f0,$f1# almaceno la suma
    addi $t1,$t1,1 #incremento del contador
    addi $t0,$t0,4 # incremento de las direcciones
    jal inicio

final:

# mostramos mensaje y la suma
li $v0, 4
la $a0, A
syscall

mov.s $f12,$f1
li $v0, 2
syscall

mtcl $a1,$f2# de entero a flotante
cvt.s.w $f2,$f2 #pasamos la longitud a estandar IEEE

div.s $f3,$f1,$f2#hacemos la media

li $a0,'\n'
li $v0,11
syscall
#mostramos mensaje y la media
li $v0, 4
la $a0, B
syscall

mov.s $f12,$f3
li $v0, 2
syscall
```

```

la suma de los elementos del vector es : 55.0
la media es: 5.5
-- program is finished running (dropped off bottom

```

Registros:

Carga el 1 de la 1ª posición y lo añade a la suma,

\$f0	1.0
\$f1	1.0
\$f2	0.0

Carga el 3 de la 3ª posición y lo añade a la suma,

Name	Float
\$f0	3.0
\$f1	6.0
\$f2	0.0

Carga el 8 de la 3ª posición del array y lo añade a la suma

Name	Float
\$f0	8.0
\$f1	36.0
\$f2	0.0

\$f1 la suma,\$f2 la longitud y \$ f3 la media

\$f0	10.0
\$f1	55.0
\$f2	10.0
\$f3	5.5
\$f4	0.0

# Practica 9

## Ejemplos

### 2. Operaciones de movimiento de datos

**2.1. ¿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?**

Porque las direcciones son enteros

**2.2. 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?**

Que ha habido un redondeo en \$f4

### 3. Instrucciones de comparación

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

Intercambio de posiciones

### 4. Saltos condicionales para coma flotante

**4.1. ¿Dónde crees que se ejecutará la instrucción bclt? ¿En la CPU o en la FPU?**

En la cpu, los flag son enteros

**4.2. ¿Qué código de condición se ve afectado? ¿Hasta cuándo permanecerá el valor del código de condición?**

El 1 ; hasta que haya una comparación

## Entregas

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

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

```
.data
A: .asciiz "la suma de los elementos del vector es : "
B: .asciiz "la media es: "
Array: .float 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
long: .word 10
Suma: .float 0
.text

la $t0, Array # cargamos la direccion del array
lw $a1, long #cargamos el tamaño
li $t1, 0 # contador
# suma de los diferentes elementos del array
inicio:
    beq $a1, $t1, final # salta si el contador es 10
    lwcl $f0, 0($t0) #carga en $f0 el elemento de la direccion de $t0
    add.s $f1, $f0, $f1 # almaceno la suma
    addi $t1, $t1, 1 #incremento del contador
    addi $t0, $t0, 4 # incremento de las direcciones
    jal inicio

final:
# mostramos mensaje y la suma
li $v0, 4
la $a0, A
syscall

mov.s $f12, $f1
li $v0, 2
syscall

mtcl $a1, $f2 # de entero a flotante
cvt.s.w $f2, $f2 #pasamos la longitud a estandar IEEE

div.s $f3, $f1, $f2 #hacemos la media

li $a0, '\n'
li $v0, 11
syscall
#mostramos mensaje y la media
li $v0, 4
la $a0, B
syscall

mov.s $f12, $f3
li $v0, 2
syscall
```

```

la suma de los elementos del vector es : 55.0
la media es: 5.5
-- program is finished running (dropped off bottom

```

Registros:

Carga el 1 de la 1ª posición y lo añade a la suma,

\$f0	1.0
\$f1	1.0
\$f2	0.0

Carga el 3 de la 3ª posición y lo añade a la suma,

Name	Float
\$f0	3.0
\$f1	6.0
\$f2	0.0

Carga el 8 de la 3ª posición del array y lo añade a la suma

\$f0	8.0
\$f1	36.0
\$f2	0.0

\$f1 la suma, \$f2 la longitud y \$ f3 la media

\$f0	10.0
\$f1	55.0
\$f2	10.0
\$f3	5.5
\$f4	0.0

**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:**

```

X = 2
n = 5
X^n = 32.0
-- program is fi

```

```

Reset: reset com

```

```

X = 9
n = 3
X^n = 729.0
-- program is fi

```

Funcion pow a completar:

```

#-----elevar-----#
pow:
li $t0, 1
inicio:

    beq $a0,$t0,final# salta si el contador es n
    mul.s $f0,$f0,$f12 # multiplicas X($f12) con la multiplicacion anterior($f0)
    addi $t0,$t0,1 #incremento del contador

    j inicio

final:
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:

funcion

```

7  #-----max-----#
8  max:
9      c.lt.s $f14, $f12#si x es mayor a y
0      bclt  xmayor #salta si x es mayor a y
1
2  jr $ra
3  xmayor: # imprime X
4      la $a0,MaxRes
5      li $v0,4
6      syscall
7      mov.s $f14,$f0
8      li $v0,2
9      syscall
0      li $v0,10
1      syscall
2      jr $ra

```

```

X = 5
Y = 1
El mayor es 5.0
-- program is finished

Reset: reset completed

X = 5
Y = 9
El mayor es 9.0
-- program is finished

```