PRACTICA 5
EJERCICIOS RESUELTOS

Procesador RISC: instrucciones de Punto Flotante y pasaje de parámetros

2) Es posible convertir valores enteros almacenados en alguno de los registros r1-r31 a su representación equivalente en punto flotante y viceversa. Describa la funcionalidad de las instrucciones **mtc1**, **cvt.l.d**, **cvt.d.l** y **mfc1**.
Del set de instrucciones:

Instrucción	***Descripción***
mtc1 r_f, f_d	Copia los 64 bits del registro entero r_f al registro f_d de punto flotante
mfc1 r_d, f_f	Copia los 64 bits del registro f_f de punto flotante al registro r_d entero
cvt.d.l f_d, f_f	Convierte a punto flotante el valor entero copiado al registro f_f, dejándolo en f_d
cvt.l.d f_d, f_f	Convierte a entero el valor en punto flotante contenido en f_f, dejándolo en f_d

Para copiar el valor que tengo en un registro entero (r0 a r31) a uno de punto flotante (f0 a f31):

- Copiar los 64 bits del registro entero rf al registro fd de punto flotante
    - **mtc1** rf, fd
- Conviertir a punto flotante el valor entero copiado al registro ff, dejándolo en fd
    - **cvt.d.l** fd, ff

Importante: los números muy grandes serán redondeados en su mejor representación de punto flotante.

Para copiar el valor que tengo en un registro de punto flotante (f0 a f31) a un registro entero (r1 a r31):

- Conviertir a entero el valor en punto flotante contenido en ff, dejándolo en fd
    - **cvt.l.d** fd, ff
- Copiar los 64 bits del registro ff de punto flotante al registro rd entero
    - **mfc1** rd, ff

Importante: El número se trunca, no se redondea.

3) Escribir un programa que calcule la superficie de un triángulo rectángulo de base 5,85 cm y altura 13,47 cm.Pista: la superficie de un triángulo se calcula como:

Superficie = (base x altura) / 2

```
        .data
base:   .double 5.85
altura: .double 13.47
sup:    .double 0.0

        .code
        l.d     f1, base(R0)
        l.d     f2, altura(R0)
        mul.d   f3, f2, f1 ; f3 = base * altura

        daddi   r2, r0, 2
        mtc1    r2, f4
        cvt.d.l f5, f4  ; f5 = 2.0

        div.d   f6, f3, f5  ; f6 = base * altura / 2
        s.d     f6, superficie(r0)
        Halt
```

8) Escriba una subrutina que reciba como parámetros las direcciones del comienzo de dos cadenas terminadas en cero y retorne la posición en la que las dos cadenas difieren. En caso de que las dos cadenas sean idénticas, debe retornar -1.

```
        .data
cadena1: .asciiz "hola"
cadena2: .asciiz "hola"
result:  .word 0

        .code
        daddi $a0, $0, cadena1
        daddi $a1, $0, cadena2
        jal   compara
        sd    $v0, result($zero)
        halt

compara: dadd  $v0, $0, $0
   loop: lbu   $t0, 0($a0)
         lbu   $t1, 0($a1)
         beqz  $t0, fin_a0
         beqz  $t1, final
         bne     $t0, $t1, final
         daddi $v0, $v0, 1
         daddi $a0, $a0, 1
         daddi $a1, $a1, 1
         j     loop
 fin_a0: bnez  $t1, final
         daddi $v0, $0, -1
  final: jr    $r
```

9) Escriba la subrutina ES_VOCAL, que determina si un carácter es vocal o no, ya sea mayúscula o minúscula. La rutina debe recibir el carácter y debe retornar el valor 1 si el carácter es una vocal, o 0 en caso contrario

```
          .data
letra:    .ascii 'O'
vocales:  .asciiz 'AEIOUaeiou'
result:   .word 0


          .code
           lbu $a0, letra($0)
           jal es_vocal
           sd  $v0, result($zero)
           halt

es_vocal:  dadd $v0, $0, $0
           daddi $t0, $0, 0
    loop:  lbu $t1, vocales($t0)
           beqz $t1, fin_vocal
           beq $a0, $t1, si_es_voc
           daddi $t0, $t0, 1
           j loop
si_es_voc: daddi $v0, $0, 1
fin_vocal: jr $ra
```

12) El siguiente programa espera usar una subrutina que calcule en forma recursiva el factorial de un número entero:

```
              .data
valor:        .word 10
result:       .word 0

              .text
              daddi $sp, $zero, 0x400      ; Inicializa puntero al tope de la pila
(1)
              ld    $a0, valor($zero)
              jal   factorial
              sd    $v0, result($zero)
              halt

factorial:    ...
              ...
              …
```

(1) La configuración inicial de la arquitectura del WinMIPS64 establece que el procesador posee un bus de direcciones de 10 bits para la memoria de datos. Por lo tanto, la mayor dirección dentro de la memoria de datos será de $2^{10} = 1024 = 400_{16}$.

a) Implemente la subrutina `factorial` definida en forma recursiva. Tenga presente que el factorial de un número entero n se calcula como el producto de los números enteros entre 1 y n inclusive:

factorial(n) = n! = n x (n-1) x (n-2) x … x 3 x 2 x 1

```
             .data
valor:       .word 10
result:      .word 0

             .code
             daddi $sp, $0, 0x400    ; Inicializa el puntero al tope de la pila
             ld    $a0, valor($0)
             jal   factorial
             sd    $v0, result($0)
             halt
```

```
factorial:  daddi $sp, $sp, -16
            sd    $ra, 0($sp)
            sd    $s0, 8($sp)
            beqz  $a0, fin_rec
            dadd  $s0, $0, $a0
            daddi $a0, $a0, -1
            jal   factorial
            dmul  $v0, $v0, $s0
            j     fin
fin_rec:    daddi $v0, $0, 1
fin:        ld    $s0, 8($sp)
            ld    $ra, 0($sp)
            daddi $sp, $sp, 16
            jr    $ra
```

b) ¿Es posible escribir la subrutina `factorial` sin utilizar una pila? Justifique.