UNIVERSIDAD CARLOS III DE MADRID. DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **2 horas y media. NO** se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.

Ejercicio 1 (2 punto). En relación al estándar IEEE 754 responda a las siguientes preguntas:

a) En una representación IEEE 754 de 32 bits, indique de forma razonada el número de valores no normalizados que se pueden representar.
b) En un computador de 32 bits, ¿Se puede representar de forma exacta el valor $2^{27}+1$ en una variable de tipo `float`? ¿y en una variable de tipo `int`? Razone su respuesta.
c) Represente en el estándar IEEE 754 de doble precisión el valor 12.5. Exprese el resultado en hexadecimal.

Solución:

a) Un valor no normalizado se corresponde con un exponente (8 bits) cuyo valor es 0 y una mantisa (23 bits) con valor distinto de cero. El número de elementos que se pueden representar es, por tanto, 2 (positivo y negativo) x 2^{23}-1 = $2(2^{23}-1)$

b) El valor $2^{27} + 1 = 10000000000000000000000000001_{(2} = 1.00000000000000000000000000001 \times 2^{27}$ (2

Dado que en la mantisa de un número de tipo `float` (IEEE 754 de 32 bits) solo se pueden almacenar 23 bits, aparte del bit implícito, para el número anterior no se podrían almacenar los 4 bits menos significativos del número, por lo que el número no se podría representar de forma exacta. En una variable de tipo `int`, que emplea complemento a 2, el rango de representación es [-2^{31}, 2^{31}-1. En este caso, sí que se puede representar el número de forma exacta.

c)

$12.5_{(10} = 1100.1_{(2} = 1.1001 \times 2^{3}{}_{(2}$

Signo = 1
Exponente = 1023+3 = $1026_{(10}$ $1024+2_{(10} = 10000000010_{(2}$
Mantisa = 1001…. 000000 (52 bits)

La representación del valor 12.5 es:

0 100 0000 0010 1001 00000000 …… 00000

Expresado en hexadecimal queda: 0x4029000000000000

Ejercicio 2 (3 puntos). Considere la rutina `Contabilizar`. Esta rutina acepta **dos** parámetros de entrada:

- Un vector de números de tipo `float`.
- El número de elementos del vector

La función devuelve **tres** valores:

- El número de elementos con valor igual a 0.
- El número de elementos correspondientes a valores normalizados distintos de 0.
- El número de elementos correspondientes a valores no normalizados (no se incluyen los valores de tipo NaN).

Se pide:

a) Codifique correctamente la rutina `Contabilizar` anteriormente descrita. Puede hacer uso de las rutinas auxiliares que considere oportuno. Ha de seguirse estrictamente el convenio de paso de parámetros y uso de pila, aunque no es necesario hacer uso del registro de marco de pila.

b) Dada la siguiente definición de vector:

```
.data
  vector: .float 0.0, 0.1, -0.2, 1.0, 1.1, 1.2, 2.0, 2.1, 2.2
```

Codifique el fragmento de código que permite invocar correctamente a la función `Contabilizar` e imprimir los valores que devuelve dicha función.

Solución:

a) Una posible implementación sería:

```
EsCero:         #comprueba si $f12 (argumento de tipo float) es 0.0
                mfc1      $t0, $f12
                beqz      $t0, true1
                li        $v0, 0          # 0: no es cero
                jr        $ra
     true1:     li        $v0, 1          # 1: es cero
                jr        $ra

EsNormalizado: #comprueba si $f12
                # (argumento de tipo float) es normalizado
                # es normalizado si:   0  < exponente < 255
                mfc1      $t0, $f12
                li        $t1,  0x7F800000       #  se  obtiene  el
exponente
                beqz      $t1, falso1
                li        $t2, 255
                beq       $t1, $t2, falso1
                li        $v0, 1          # 1: es normalizado
                jr        $ra
     falso1:    li        $v0, 0          # 0: no es normalizado
                jr        $ra
```

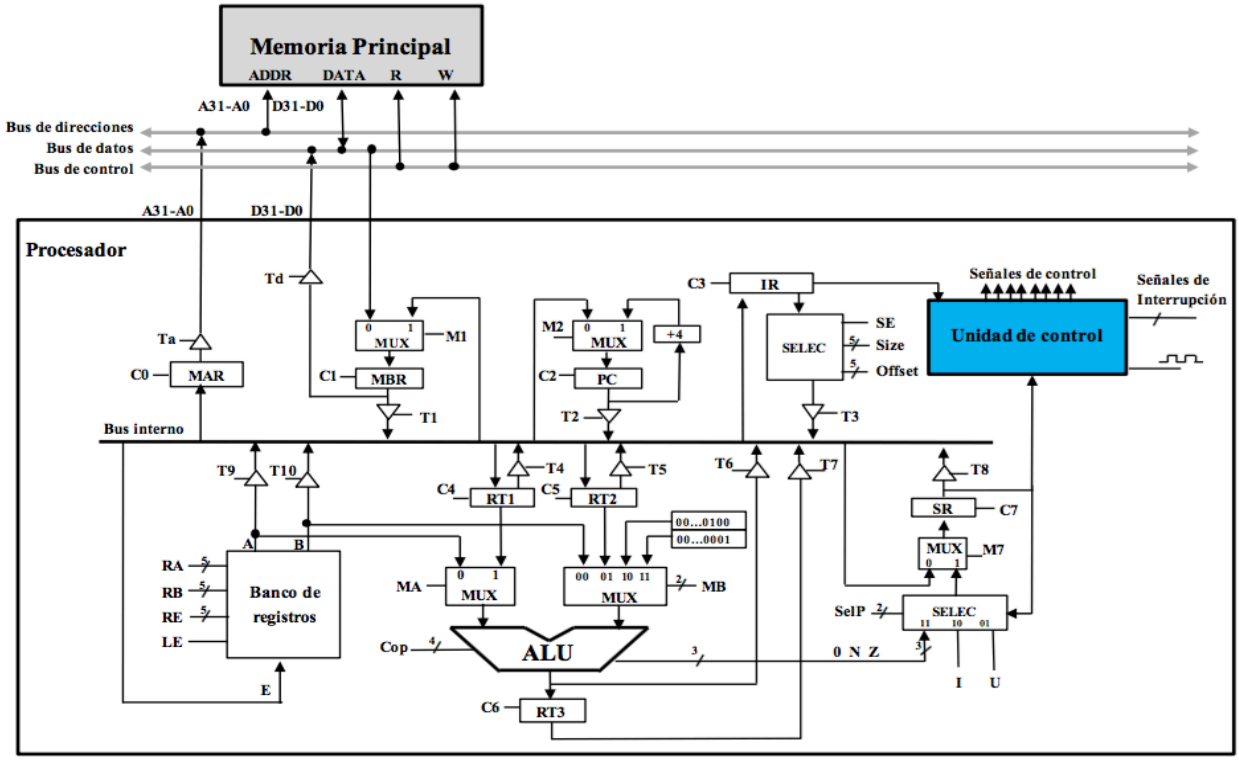
```
EsNoNormalizado:    #comprueba si $f12 (argumento de tipo float)
                    # es no normalizado
                # exponenete igual a 0 y mantisa distinto de 0
                mfc1      $t0, $f12
                li        $t1, 0x7F800000  # se obtiene el exponente
                beqz      $t1, comprueba
                li        $v0, 0    #0: no es no normalizado
                jr        $ra
comprueba:      li        $t1, 0x007FFFFF # se obtiene la mantisa
                beqz      $t1, falso2
                li        $v0, 1    # 1: es no normalizado
                jr        $ra
     falso2:    li        $v0, 0    # 0: no es no normalizado
                jr        $ra


Contabilizar:   addi      $sp, $sp, -20
                sw        $s0, 16($sp)
                sw        $s1, 12($sp)  # número de ceros
                sw        $s2,  8($sp)  # número de normalizados
                sw        $s3,  4($sp)  # número de no normalizados
                sw        $ra,   ($sp)

                li        $s0, 0    # índice del bucle
                li        $s1, 0    # inicializar número de ceros
                li        $s2, 0    # inicializar número de norm.
                li        $s3, 0    # inicializar número de no norm.


     bucle:     bgt       $s0, $a1, fin
                l.s       $f12, ($a0)    # el siguiente elemento
                                         # del vector

                jal       EsCero
                addi      $s1, $s1, $v0

                l.s       $f12, ($a0)
                jal       EsNormalizado
                addi      $s2, $s2, $v0

                l.s       $f12, ($a0)
                jal       EsNoNormalizado
                addi      $s3, $s3, $v0

                addi      $s0, $s0, 1
                addi      $a0, $a0, 4    # preparar el siguiente
                                         # elemento del vector
                b         bucle

     fin:       move      $v0, $s1  # número de ceros
                move      $v1, $s2  # número de normalizados
                move      $t0, $s3  # número de no normalizados
```

```
        lw      $s0, 16($sp)
        lw      $s1, 12($sp)
        lw      $s2,  8($sp)
        lw      $s3,  4($sp)
        lw      $ra,   ($sp)

        # número de no normalizados en la cima de la pila
        addi    $sp, $sp, 16
        sw      $t0, ($sp)
        jr      $ra
```

b) El fragmento necesario para invocar a la función anterior es:

```
# se pasan los parámetros de entrada
la      $a0, matriz
li      $a1,12

jal     Contabilizar

# primer resultado en $v0, segundo en $v1 y
# el tercero en la cima de la pila
move    $a0, $v0
li      $v0, 1
syscall      # se imprime el número de ceros

move    $a0, $v1
syscall      # se imprime el número de normalizados

# se extrae el tercer resultado de la cima de la pila
lw      $a0, ($sp)
addi    $sp, $sp, -4
syscall      # se imprime el número de no normalizados
```

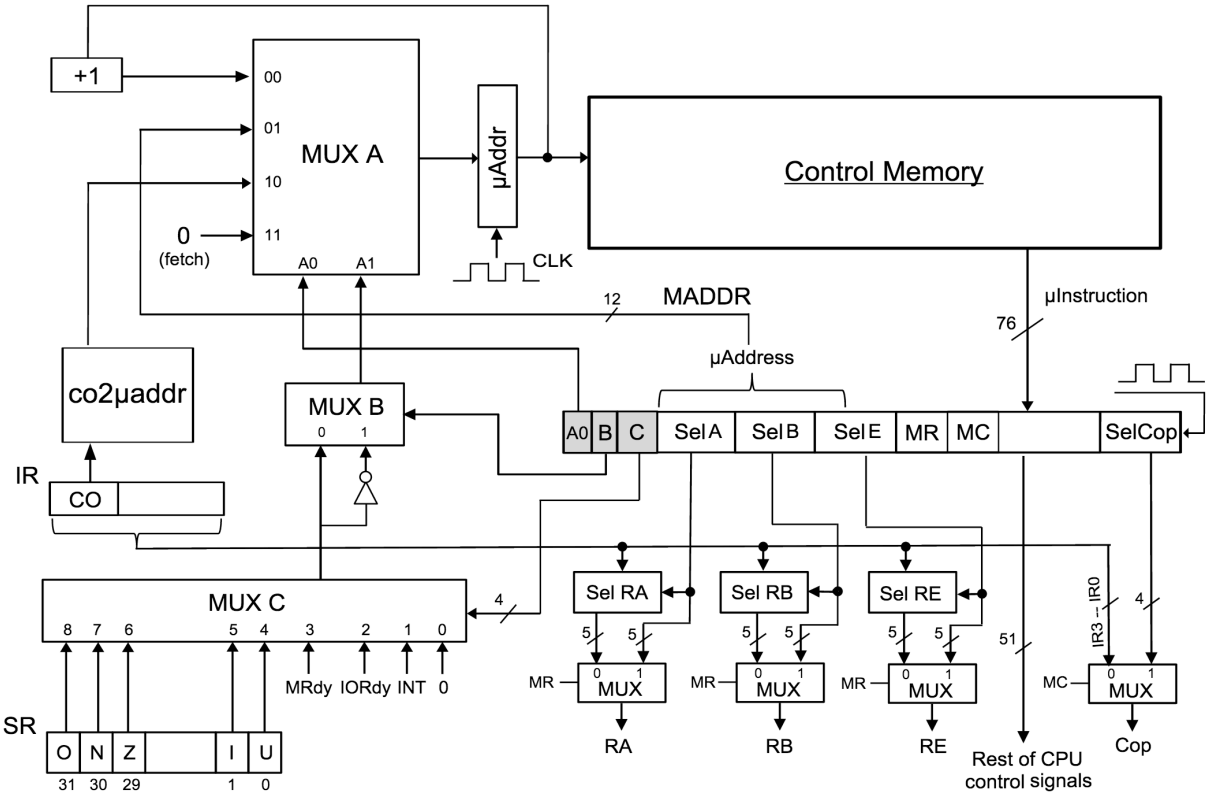
Ejercicio 3 (3 puntos). Dado el procesador con la siguiente estructura:



Y con las siguientes características:

- Un banco de registros con 32 registros de 32 bits, capaz de procesar 127 instrucciones distintas.
- Una ALU con 16 operaciones aritmético-lógicas que incluyen la suma, resta, multiplicación, los and, or, not, xor a nivel de bit, la rotación a la derecha e izquierda de un bit, y los desplazamientos lógicos y aritméticos a la derecha, y el desplazamiento lógico a la izquierda.
- Memoria direccionable a nivel de byte y un solo ciclo de reloj para la lectura o escritura.

Este procesador dispone de una Unidad de control representada por la siguiente figura:



Responda de forma breve y justificada a las siguientes preguntas:

a) Analizando el diagrama de la Unidad de Control, ¿Qué **técnica de control** emplea? ¿Qué **tipo de secuenciamiento** usa?

b) Dada la instrucción de dos palabras **`li.w R valor`** que almacena "valor" en el registro "R", indique un formato de instrucción para la misma de forma que en la primera palabra contenga el registro "R" y en la segunda palabra esté "valor" como número binario de 32 bits.

c) Especifique las operaciones elementales y todas las señales de control necesarias, tanto de la unidad de control como del procesador, para ejecutar la instrucción máquina **`li.w R valor`** anteriormente descrita. Incluya el ciclo de *fetch*.

Operaciones elementales	**Señales de control**
…	…

NOTA: Asuma que R29 actúa como puntero de pila, que el puntero de pila apunta a cima de pila y que la pila crece hacia direcciones decrecientes de memoria.

Solución:

a) Esta Unidad de Control tiene una técnica de **control microprogramado** y un **secuenciamiento implícito**.

b) Un posible formato para la instrucción descrita sería:

Primera palabra:

Código de instrucción **(7 bits)**	Registro **(5 bits)**

Segunda palabra:

Valor **(32 bits)**

c) Las operaciones elementales y las señales de control:

Operaciones elementales	Señales de control
fetch	
MAR <- PC	T2, C0
MBR <- Mem[MAR]	TA, R, BW=11, C1=1
PC <- PC+4, IR <- MBR	M2, C2, T1, C3
Salto a c.o.	A0, B=0, C=0
li.w R valor	
MAR <- PC, PC <- PC+4	T2, C0, M2=1, C2
MBR <- Mem[MAR]	TA=1, R=1, BW=11, C1=1
R <- MBR, salto a fetch	T1=1, LE=1, MR=0, SELE=10100, A0=1, B=1, C=0

Ejercicio 4 (2 puntos). Considere el siguiente fragmento de código:

```
double A[10000];

for (i=0; i<10000; i++) {
     A[i] = A[i] + 3;
}
```

Dicho código se ejecuta en una arquitectura con un ancho de palabra de 32 bits, que incluye un sistema de memoria virtual que emplea páginas de 8 KB.
Se pide:

a) Indique el **formato de la dirección virtual** y describa el contenido de una **entrada de la tabla de páginas**.
b) Asumiendo que inicialmente no hay ninguna página en memoria principal y que los datos e instrucciones se almacenan en páginas distintas, indique el número de fallos de página que se producen durante la ejecución del fragmento de programa anterior y la **tasa de aciertos** considerando solo las páginas de datos que almacenan los datos del vector `A`.

Solución:

a) El formato de la dirección virtual con páginas de 8 KB (13 bits de desplazamiento) en una máquina de 32 bits es:

32 – 13 = **19 bits**	**13 bits**
Id. de página	Desplazamiento

El contenido típico de una entrada en la tabla de páginas es:

P	M	RWX	Marco
Bit de presente	Bit de modificado	Bits de permisos	Bits para indicar el marco asociado

b) Se empezará por el análisis de la imagen del proceso:

El vector A ocupa 10 000 números decimales en doble precisión (64 bits cada uno). Por tanto precisará de 10 000 × 8 = 80 000 bytes y usando páginas de 8 KB este vector necesita de 80 000 / 8192 = 9,766 páginas. Redondeando por exceso, precisa de 10 páginas de 8 KB.

En el segmento de código estará el bucle `for`, lo que supone una página de 8 KB como mucho para almacenar el código binario asociado.

A continuación se analizará la traza de acceso a los datos del vector A: durante cada iteración del bucle `for` se producen dos accesos, uno de lectura y uno de escritura, por tanto se producen 20 000 accesos. Dado que el vector se procesa de forma secuencial, se producirá un fallo cada vez que se accede al primer elemento de una página. Como el vector A ocupa 10 páginas, se producirán 10

fallos de página.,

La tasa de aciertos, por tanto será: (20 000-10) / 20 000 = 0.9995. Es decir, la tasa de aciertos será del 99,9 %.