Ejercicio 1 – Representación de la información

Sean los siguientes formatos de representación de números decimales:

a) `bfloat16` *(Punto flotante de IEEE 754 acortado de 32 a 16 bits)*:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Signo	exponente (e)								mantisa (m)						

Las reglas de codificación son las mismas de IEEE 754 32 bits acortando la mantisa a 8 bits.

b) `fixed16` *(punto fijo de 16 bits)* con la siguiente distribución de bits:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Signo	parte entera										parte decimal				

1. Indicar en decimal para cada formato propuesto el valor del mayor número representable, el menor y el positivo más cercano a cero. Dar las representaciones en binario de cada uno.
2. Dada la siguiente cadena de bytes: `AA 05 BC 00 7F AA`, interpretar 3 números contiguos, suponiendo primero que están codificados en `bfloat16` y luego que están codificados en `fixed16`.
3. Dar y justificar 3 ejemplos de operaciones aritméticas concretas en los cuales el resultado de la operación sea a) en `bfloat16` más preciso que `fixed16`; b) que ocurra el caso inverso y c) tenga la misma precisión en ambos formatos.
4. Codificar, si es posible, los siguientes números en decimal en el formato del enunciado. Redondear en caso de que no sea posible una representación exacta. Indicar el error de representación en cada caso.[1]
   a) 115360 *b*) −0,1875 *c*) 248,4
5. ¿Existe una única representación para cada número expresable en cada formato propuesto? Justificar.

Ejercicio 2 – Lógica digital

Resolver los siguientes circuitos.

1. Construir un:
   a) Registro `Paralelo-Paralelo` de 6 bits.
   b) Contador de 2 bits, ascendente, satura en valor 3 (sin volver a cero).
   c) Sumador de 6 bits, utilizando sumadores completos.
2. Construir un circuito de 3 entradas (`A`, `B`, `C`) de 6 bits y una salida de 6 bits (`S`), tal que realice la operación `(A+B+C)/2`, considerando las entradas como números en complemento a 2. Adicionar una salida que indique si la operación resultó en un número entero exacto y otra que indique si se produjo *overflow* en el cálculo.
3. Construir un circuito secuencial que calcule términos de la serie numérica $F_0 = x_0, F_1 = x_1, F_2 = x_2, F_{n+3} = \frac{F_n + F_{n+1} + F_{n+2}}{2}$. El circuito debe permitir la carga secuencial de los valores iniciales x_0, x_1, x_2 de la sucesión a registros correspondientes. Es necesario haber cargado 3 valores iniciales antes de que el circuito pueda comenzar a calcular y emitir una salida. Una vez cargados al menos 3 valores se empieza a calcular términos utilizando el circuito anterior y almacenando el resultado más recienteen un registro de salida. Cada

[1] Se entiende como error de representación al valor absoluto de la diferencia entre el valor codificado y el original.

nuevo termino debe retroalimentarse a los registros de entrada. Este circuito debe estar conectado a un bus de datos de 6 bits y contará con las señales de control en que permite la entrada de datos al circuito, `enOut` que permitirá exhibir el resultado a la salida (siempre y cuando haya podido calcular al menos un valor) y la entrada de reloj. Las salidas que indican si el resultado es entero o si hubo overflow deben ser almacenadas en registros de 1 bit. Considerar que el registro resultado solamente será modificado cuando se haya finalizado la carga de los valores iniciales y se haya deshabilitado la entrada de datos.

Nota: Se puede utilizar compuertas `AND`, `OR`, `XOR` y `NOT`. Además de biestables, buffer de tres estados o cualquier circuito ya construido en el ejercicio.

Ejercicio 3 – Seguimiento

Considerar el siguiente código cargado a partir de la dirección `0xFFE0`.

```
main:   CALL start
        DW 0xDEAD
start:  SUB R0, 0x0001
        MOV R1, [0xFFEF]
        JLEU skip
        ADD [R1+0x000C], [0xFFE9]
        JE start
        RET
skip:   JNE main
        CALL 0xFFE4
```

1. Calcular las posiciones de las etiquetas y los desplazamientos de todos los saltos condicionales.
2. Realizar el seguimiento del código hasta encontrar una instrucción inválida.
3. Listar las posiciones de memoria que fueron modificadas durante la ejecución y los valores escritos.

Ejercicio 4 – Formato de instrucción

Considerar el siguiente conjunto de instrucciones, para un procesador con 16 registros de propósito general de 16 bits `r0` a `r15`. La memoria de datos y la de instrucciones es compartida, y tiene direcciones de 16 bits y direccionamiento a palabras de 16 bits.

Instrucción	Acción
`add rA, rB, rC`	`rA` ← `rB + rC`
`addi rA, rB, imm4`	`rA` ← `rB + imm4` (inmediato de 4 bits en complemento a 2)
`nand rA, rB, rC`	`rA` ← ∼(`rB & rC`)
`or rA, rB, rC`	`rA` ← `rB | rC`
`xor rA, rB, rC`	`rA` ← `rB ^ rC`
`lui rA, imm8`	`rA[15..8]` ← `imm8` (inmediato de 8 bits sin signo); `rA[7..0]` ← 0
`lli rA, imm8`	`rA[7..0]` ← `imm8` (inmediato de 8 bits sin signo)
`sw rA, rB, imm5`	`rA` ← `[rB + imm5]` (inmediato de 5 bits sin signo)
`lw rA, rB, imm5`	`[rB + imm5]` ← `rA` (inmediato de 5 bits sin signo)
`beq rA, rB, imm6`	Si `rA = rB`, `PC` ← `PC + imm6` (inmediato de 6 bits en complemento a 2)
`jalr rA, rB`	`rA` ← `PC`; `PC` ← `rB`
`halt`	Detiene la ejecución.

1. Identificar los tipos de instrucciones a codificar (cantidad y tipos de operandos) y la cantidad de cada tipo.
2. Para las instrucciones mencionadas, diseñar un formato de instrucción de tamaño fijo de 16 bits.
3. Calcular cuántas instrucciones nuevas sin operandos pueden ser codificadas. ¿Y cuántas que tomen dos registros?
4. La instrucción `addi` toma dos registros y un inmediato de 4 bits en complemento a 2. ¿Es posible extender el lenguaje ensamblador de esta máquina para que permita sumar inmediatos de 8 bits, también en complemento a 2? En caso afirmativo, explicar cuál sería el mecanismo.