deben ser incluidas en el examen.

| 1 | 2 | 3 | 4 | |
|---|---|---|---|---|
| B- | R- | M | B | A |

CORRECTOR: MAXI

**Ejercicio 1** Representación de la información

Sea el siguiente formato de números de punto floante ($s$ indica el bit de signo de la mantisa):

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| s | exponente | | | | mantisa | | |

Y las siguientes reglas de codificación:

- Exponente en notación exceso 2.
- $E_{min} = -1$; $E_{max} = 13.$ ~~13~~ 12
- Las reglas de las formas normalizadas y denormalizadas, ceros, infinitos y `NaN` son iguales a las de `IEEE 754` precisión simple.

1. Indicar en decimal el valor del mayor número representable, el menor y el positivo más cercano a cero. Dar las representaciones en binario de cada caso.

2. Dada la siguiente cadena de 4 bytes: `33 CC FA 03` interpretar 4 números contiguos, sabiendo que cada uno de ellos se encuentra codificado en el formato del enunciado.

3. Codificar los siguientes números en el formato propuesto. Redondear en caso de que no sea posible una representación exacta. Indicar el error de representación en cada caso[1].
   *a*) $2{,}55 \times 10^{-1}$ *b*) $-4{,}5$ *c*) 240

4. Comparar este formato con el formato *complemento a 2 de 8 bits*. Justifique sus respuestas.
   *a*) ¿Qué formato representa más números distintos? Dar la cantidad de números distintos representables de cada formato.
   *b*) ¿Qué formato tiene un mayor *rango numérico* representable?
   *c*) Dar ejemplos de números *dentro del rango de representación de ambos formatos* que sean representables de manera exacta en un formato y no en el otro y vice versa.

5. ¿Existe una única representación para cada número expresable en el formato propuesto? Justificar.

**Ejercicio 2** Lógica digital

Utilizando circuitos sumadores o substractores, registros de $n$ bits y compuertas o multiplexores según se requiera:

1. Construir un circuito secuencial *inversor* con entradas A,B y salidas X,Y de 8 bits, que mediante una línea de control intercambie o no los valores en la salida.

2. Construir un circuito secuencial *3-ordenador* con entradas A,B,C y salidas X,Y,Z de 8 bits sin signo, tal que en las salidas X,Y,Z se observen los valores A,B,C ordenados de mayor a menor.

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

3. Construir un circuito secuencial con una entrada y una salida de datos (con sus correspondientes *enable*) de 8 bits sin signo que almacene internamente los 2 valores más grandes observados históricamente en la entrada, y el mínimo histórico observado. En cada ciclo (si la entrada está habilitada) leerá de la entrada un nuevo valor y actualizará el ranking de los 2 mejores y el mínimo global. El circuito debe proveer una línea de 2 bits de control que determine si en la salida se mostrará el máximo, el segundo puesto, el mínimo global, o la entrada actual del circuito. Puede asumir valores iniciales de los registros en caso de necesitarlo.

*Sugerencia: reutilice componentes ya diseñados todo lo que pueda.*

**Ejercicio 3** Programación en assembler

En una máquina ORGA1, se tiene una cadena de caracteres de longitud indeterminada a partir de la posición de memoria `0x1000`. Los mismos representan caracteres codificados en UTF-8, donde cada palabra indica 16 bits consecutivos de la cadena en *big endian*. Se indica la finalización de la cadena con el valor `0x00`[2]. Se pide:

1. Implementar en assembler de ORGA1 la función `contarUTF8`, que dada la cadena de caracteres indicada, devuelva en el registro `R0` cuántos caracteres UTF-8 tiene. Puede suponer que la cadena está bien formada (no hay caracteres inválidos o incorrectamente codificados) y que todos ocupan exactamente 1 ó 2 bytes.

   Por ejemplo, para las siguientes cadenas de palabras de 16 bits en hexadecimal[3]:

   - `3A3B 0000` debe devolver 2.
   - `5020 3BC2 80DF BF00` debe devolver 5.
   - `7F00` debe devolver 1.

   **Ayuda:** Un caracter UTF-8 de 1 byte tiene la forma `0xxx xxxx`, mientras que uno de 2 bytes tiene la forma `110x xxxx 10xx xxxx`. Separe en casos según el caracter comience en la parte más significativa o en la menos significativa de la palabra.
2. Resolver etiquetas y saltos condicionales del código escrito en el ítem anterior, sabiendo que el mismo se ensambla a partir de la posición `0x2000`.

**Ejercicio 4** Formato de instrucción

Considerar el siguiente conjunto de instrucciones, llamado VeryRISC, para un procesador con 7 registros de propósito general de 8 bits `r0` a `r6`. La memoria de datos y la de instrucciones es compartida, y tiene direcciones de 8 bits y direccionamiento a palabras de 8 bits.

| Instrucción | Acción |
|---|---|
| `inc rA` | `rA` ← `rA + 1` |
| `dec rA` | `rA` ← `rA - 1` |
| `beq imm4` | Si `Z`, `PC` ← `PC + imm4` (inmediato de 4 bits en complemento a 2) |
| `jmp imm4` | `PC` ← `PC + imm4` (inmediato de 4 bits en complemento a 2) |
| `jmp rA` | `PC` ← `rA` |
| `add rA, rB` | `rA` ← `rA + rB` (no cambia flags) |
| `sub rA, rB` | `rA` ← `rA - rB` (no cambia flags) |
| `cmp rA, rB` | Cambia flags según `rA - rB` |
| `lui imm4` | `r0[7..4]` ← `imm4` (inmediato de 4 bits sin signo) |
| `lli imm4` | `r0[3..0]` ← `imm4` (inmediato de 4 bits sin signo) |
| `mov rA` | `rA` ← `r0` |
| `ld rA` | `r0` ← `[rA]` |
| `str rA` | `[rA]` ← `r0` |

1. Identificar los tipos de instrucciones a codificar (cantidad y tipos de operandos) y la cantidad de cada tipo.
2. Diseñar un formato de instrucción de tamaño fijo de 8 bits para VeryRISC.
3. Calcular cuántas instrucciones nuevas sin operandos pueden ser codificadas.
4. La arquitectura propuesta no cuenta con una instrucción `mov rA, rB`. Sin modificar el set de instrucciones, ¿es posible extender el lenguaje ensamblador de esta máquina para que permita realizar dicha operación? En caso afirmativo, indicar cuál sería el mecanismo.

[2] Notar que este valor puede aparecer tanto en la parte alta como en la parte baja de una palabra, según dónde haya terminado el caracter anterior.

[3] La tonalidad alternada indica los límites entre caracteres.