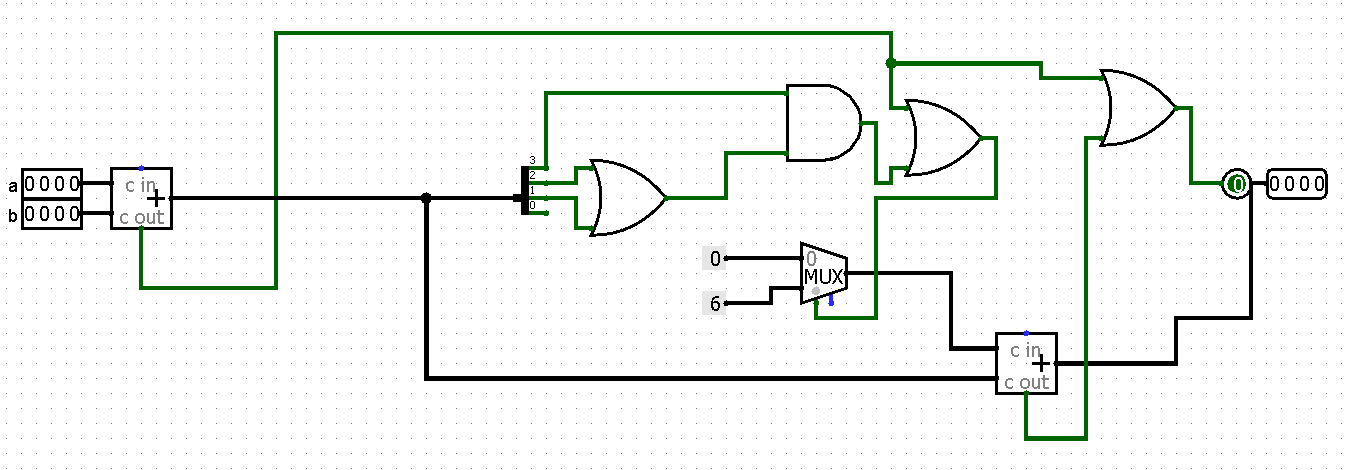
**BOLETÍN 2. PRACTICAS DE SISTEMAS DIGITALES COMBINACIONALES UTILIZANDO MODULOS.**

**1. [Síntesis modular de un circuito aritmético] Diseñar un sistema que reciba como entradas dos números sin signo A y B de 4 bits cada uno y codificados en BCD y proporcione como salida la suma de ambos números también en BCD. Para la realización de este diseño no se pueden utilizar los módulos comparadores de números binarios disponibles en la biblioteca de Logisim.**



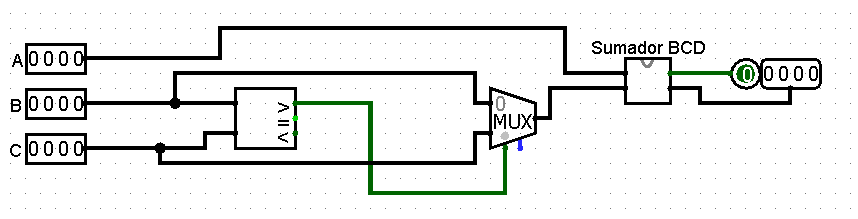
**Descripción del circuito:**

Ambos números se suman mediante un sumador. El resultado de esta suma luego se hace pasar por otro sumador, que le sumará 0 o 6.

Le sumará 6 si se cumple la condición del multiplexor: que el número resultante sea mayor de 10 en BCD: es decir, el bit más significiativo del número sea 1 y uno de los dos siguientes también sea 1.

El sistema determina también el bit más significativo del resultado, que tiene hasta 5 bits. Este bit será 1 si se cumple una de dos condiciones: si hay acarreo en la suma inicial (el bit más significativo de ambos sumandos es 1) o si hay acarreo en la suma de 6.

**2. [Síntesis modular de un circuito.] Realizar un circuito que dados los números A, B y C enteros positivos cada uno de cuatro bits y codificados en BCD, realice la siguiente función F sobre ellos: F = A + mín{B, C}. Utilizar para realizar este diseño el módulo diseñado en la práctica anterior y los módulos y puertas lógicas adicionales que consideres oportuno.**

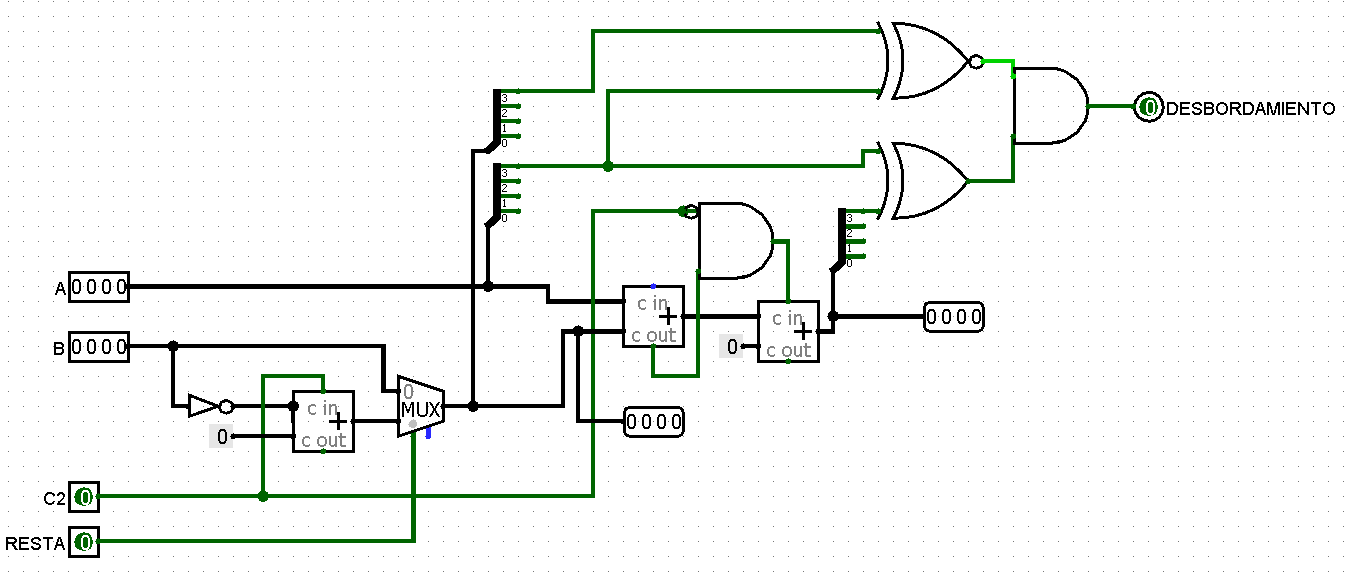


**Descripción del circuito:**

La entrada A pasa directamente al sumador BCD, que es el circuito descrito en el anterior ejercicio.

Las entradas B y C se hacen pasar por un multiplexor que toma el comparador (B>C) como condición. Si no se cumple, deja pasar B, que es menor. Si sí se cumple, deja pasar C.

**3. [Diseño de un módulo sumador en C1 y C2 a partir de un módulo sumador binario de 4 bits y de puertas lógicas.] Realizar un sumador/restador para números de 4 bits codificados en complemento a 1 (C1) o complemento a 2 (C2). El usuario podrá seleccionar si trabaja en C1 o C2. Las entradas se interpretarán como números en el formato que haya elegido el usuario y la salida estará en el mismo formato. El usuario también podrá seleccionar si quiere realizar suma o resta sobre los operandos de entrada. Las salidas del módulo sumador/restador serán los bits del resultado y un bit que indique si se ha producido overflow (desbordamiento). Realizar el diseño más simple que sea posible teniendo en cuenta que no se pueden utilizar los módulos restadores disponibles en logisim. El módulo debe realizar las operaciones en el formato que seleccione el usuario, es decir, no es válido, por ejemplo, transformar los operandos a C1 en la entrada, operar en C1 y transformar el resultado al formato que requiera el usuario.**



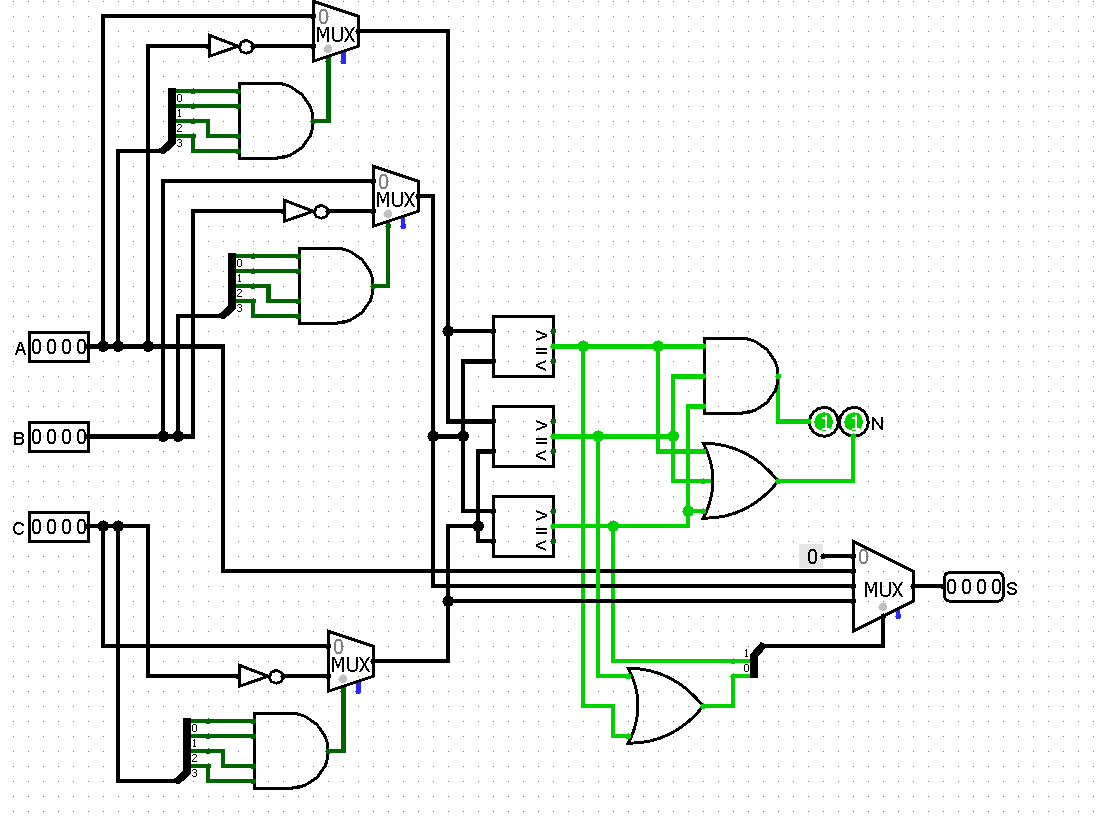
El circuito toma como entradas los dos sumandos y la selección de C1 (0) o C2 (1), además de si la operación es una suma(0) o resta(1).

Si la operación es una resta, en lugar de la entrada B se toma su inversa, mediante un multiplexor que tiene por condición la entrada (RESTA). Si, además, es un número en C2, al hacer la inversa se le suma 1. (este número se añade como acarreo en la suma).

Luego, se suman los números. Si se cumple que hay un acarreo final y la operación NO está en C2 (es decir, está en C1), se suma 1 más al número, pues el C1 el acarreo final se suma al bit menos significativo.

El circuito también comprueba si se produce desbordamiento en la operación. Se produce desbordamiento cuando los dos números a sumar/restar tienen el mismo signo, pero este signo es distinto que el del resultado. Estas comparaciones se realizan mediante puertas XOR en la parte superior.

**4. [Síntesis de un módulo comparador a partir módulos combinacionales y puertas lógicas.] Construir un dispositivo con entradas A, B y C de cuatro bits que son números en C1. El sistema tiene una salida S de 4 bits y una salida N de dos bits de modo que: La salida S será 0 si las entradas A, B y C (de 4 bits cada una) son todas diferentes entre sí. La salida S será igual a la entrada repetida en otro caso. Por otro lado, la salida N indica cuantas entradas son iguales entre sí.**



Debido a que los números están codificados en CA1, debemos tener en cuenta que 1111 = 0000. Para esto, hacemos un caso separado para cada número, donde si todos sus bits son 1, se toma su negación (0000) en su lugar para las comparaciones. Esto sec umple con un separador que lleva a una puerta AND de 4 entradas, que sirve como multiplexor que escoge entre el número y su negación.

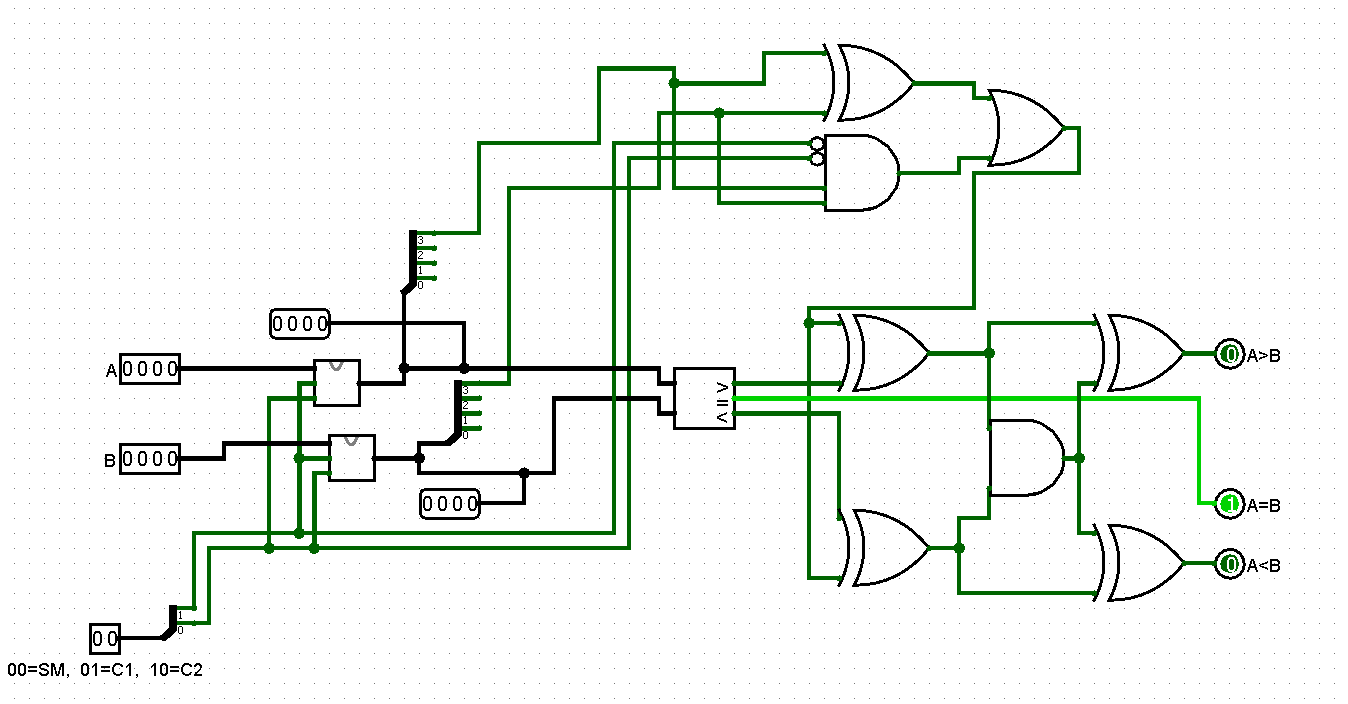
Los números, negados si es necesario, pasan a una serie de 3 comparadores que comparan si A=B, A=C y B=C. Para la salida N hay dos posiblidades: que se repita 1 número (01) y que se repitan 3 (11). Entonces, el segundo bit será 1 si al menos una comparación se cumple (puerta OR) y el primero será 1 si se cumplen todas las comparaciones (puerta AND).

Para la salida S utilizamos un multiplexor que toma 4 entradas: el 0 y las entradas A,B,C. Como condición toma 2 bits, la comparación B=C y una puerta OR entre las comparaciones A=C y A=B. De esta forma:

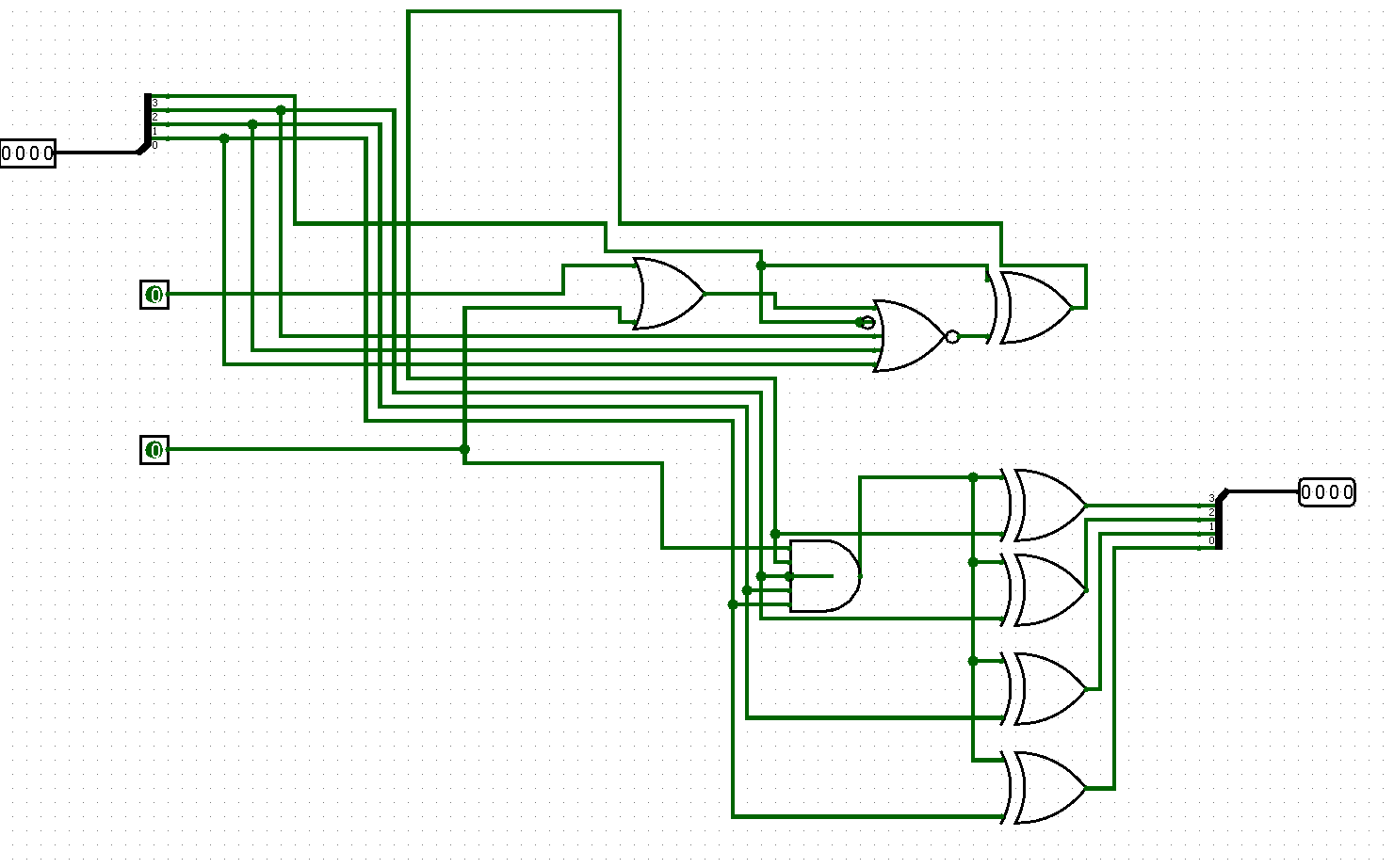
* Si no se cumple ninguna condición (00), se pasa el número 0000.
* Si se cumple A=B o A=C pero no B=C (01), pasa el número A.
* Si se cumple B=C pero no A=B ni A=C (10), pasa el número B.
* Si se cumple A=B=C (11), pasa el número C.

Entonces, siempre pasa un número que se repite.

**5. [PRACTICA OPTATIVA. Utilización de módulos combinacionales para rea- ´ lizar comparadores.] Dise˜nar un circuito comparador de dos números A y B de 4 bits cada uno, incluido el bit de signo, y codificados ambos en complemento a 1 (C1), complemento a 2 (C2) o signo magnitud (SM), y los bits necesarios para indicar si los números de entrada están los dos en SM, C1 o C2. A y B están codificados en la misma codificación. El sistema proporciona 3 bits de salida, uno para indicar A > B, otro para indicar si A < B y otro para indicar A = B. NOTA: Para realizar este dise˜no no es válido para comparar los números pasarlos a binario sino que se deben comparar directamente en SM, C1 o C2 según corresponda. Se valorará especialmente que el dise˜no sea lo más sencillo que sea posible.**

****

Antes de comparar los números, el sistema los pasa por un módulo diseñado con la tarea de corregir las excepciones que existen para el 0 en estas codificaciones.



Este programa toma un número de 4 bits y la codificación en la que está. Si está en C1 y el número es el 1000, se niega el primer bit mediante un XOR, transformando el número en el 0000. Además, si está en SM y el número es el 1111, se niegan todos los bits con XOR para transformar el número en el 0000. De esta forma, corregimos los errores que pueden existir en ambas codificaciones sin utilizar ningún multiplexor, que resultaría en un circuito más complejo.

Volviendo al sistema inicial, los números se separan y se toma su bit del signo para compararlo, mientras que el número entero se lleva a un comparador. El resultado de este comparador, sin embargo, deberá ser invertido en ciertos casos:

* Si el signo de los números es distinto, en cualquier codificación (XOR entre los bits de signo)
* Si ambos números son negativos Y la codificación es SM. (AND con ciertas entradas negadas)

Para encontrar estos casos, realizamos un OR entre ambas puertas. Luego, la salida de este OR se une con la salida del combinador utilizando un XOR, de tal forma que la salida quedará negada si es necesario.

Sin embargo, esta forma de realizar el sistema devolverá un error cuando es necesario negar el resultado pero los números son iguales (ejemplo: 1100 y 1100 en SM), pues en este caso se activarán las salidas de <, > y =. Para corregir este caso específico se un AND entre las salidas < y > y un XOR, de tal forma de que, si < y > son 1 simultáneamente, ambas se vuelve a negar.

Esta forma de realizar el sistema permite evitar por completo el uso de multiplexores.

**6. [PRACTICA OPTATIVA. Dise˜no de circuitos aritméticos.] ´ Un Consejo de Administración está constituido por: un presidente, un secretario y tres consejeros. En las votaciones, el voto del presidente tiene peso 4, el secretario peso 3 y cada uno de los votos de los consejeros peso 1. Para aprobar (1 lógico) o denegar (0 lógico) una propuesta, se suman los votos con su correspondiente peso. Si resultase empate, se resolverá por mayoría de votos (suma simple, sin tener en cuenta los pesos anteriores). Realizar el dise˜no del sistema que decida si se acepta una propuesta (salida a 1) o se rechaza (salida a 0) en función de los votos de los miembros del Consejo a partir de los siguientes módulos y utilizando el hardware adicional que sea necesario: Un módulo que sea un sumador completo de 1 bit (full adder), es decir, un módulo que suma 3 bits de entrada del mismo peso (2 operandos y un bit de acarreo) y produce dos bits de salida. Un módulo que, utilizando únicamente sumadores completos como el descrito en el apartado anterior, sume los votos del Consejo de Administración con su peso. Un módulo que, utilizando también solamente bloques consistentes cada uno en un sumador completo, sume el número de votos afirmativos sin incluir el peso.**

**7. [PRACTICA OPTATIVA. Utilización de módulos combinacionales comparadores y sumadores.] Para entrar en un recinto hay que pasar dos puertas P1 y P2. para entrar por P1 hay que introducir un dígito BCD mayor o igual que 6 y para pasar por la puerta P2 hay que introducir un número de 4 bits que esté entre 3 unidades por arriba y por abajo de la mitad del número introducido en la puerta P1. Por ejemplo, si introducimos el 8, entramos por P1 y debemos meter un número entre 4 − 3 y 4 + 3 para entrar por P2. Dise˜nar el circuito lógico correspondiente usando módulos combinacionales y puertas lógicas.**