LÓGICA BINARIA BÁSICA

02

1.- OPERADOS LÓGICOS.

Los operadores lógicos son funciones de verdad. Quiere esto decir que son funciones que toman, en su forma básica, uno o dos valores de verdad (entradas, verdaderas o falsas), y devuelven un único valor de verdad (salida, verdadera o falsa). En consecuencia, cada conectiva lógica (operador) puede ser definida mediante una tabla de valores de verdad que indique qué valor devuelve el operador para cada combinación de valores de verdad.

Esos operadores lógicos son la base de cualquier circuitería digital. La implementación física de estos operadores lógicos, en los circuitos electrónicos, se denominan "puertas lógicas" y se representan, en los esquemas de los circuitos, por los símbolos que se indicarán en cada caso de acuerdo con las normativas **ASA**[1] (***A**merican ES**t**andars **A**ssociation*, Asociación de Estándares Estadounidense) y **DIN**[2] (***D**eutsches **I**nstitut für **N**ormung*, Instituto Alemán de Normalización) por ser las más utilizadas, aunque no las únicas.

Estos mismos operadores lógicos son de uso habitual en programación, en general, y son básicos, por ejemplo, a la hora de plantear consultas a cualquier base de datos, tal como obtener el listado de todos los usuarios cuyo nombre sea José **Y (and)** el primer apellido García **Y (and)** el segundo apellido Pérez **O (or)** Vázquez), consulta qué podríamos esquematizar como: José **Y** García **Y** (Pérez **O** Vázquez), en donde se evidencia el uso de los operadores lógicos y su función conectiva.

En redes, también se utilizan de forma habitual. Por ejemplo, gracias a ellos (operador **and**) es posible saber a qué subred pertenece una IP, conociendo su máscara de subred.

Estas puertas lógicas suelen utilizarse agrupadas (en cascada) de forma que la salida de una de ellas corresponda a una de las entradas de otra, y así sucesivamente, hasta completar circuitos que pueden realizar operaciones realmente complejas de forma extraordinaria eficaz, tal y como se ve en el ejemplo Figura 9. Adviértase que estas puertas lógicas son los componentes mínimos funcionales, aunque no en exclusiva, de los microprocesadores que hacen funcionar los ordenadores actuales.

Estos operadores están sometidos al álgebra de proposiciones[3], lo que permite un estudio matemático y formal de los mismos. Facilitando el desarrollo y simplificación (optimización) de funciones lógicas compuestas, extraordinariamente complejas, tanto desde el punto de vista formal como funcional.

Como ya se comentó, para el análisis de las funciones lógicas se utilizan las "tablas de verdad", que se construyen disponiendo en distintas columnas todas las posibles entradas y sus correspondientes salidas, obteniéndose unas tablas en las que se muestran todas las posibles combinaciones de los valores de las entradas del sistema y el valor generado, por la función lógica, para cada una de las entradas posibles, como salida.

Para entender las tablas de verdad, desarrollaremos un pequeño ejemplo de la vida cotidiana. Admitamos que salimos de casa con la firme decisión de comprar un pantalón vaquero azul. Lo primero que tenemos que hacer es formalizar nuestra decisión de compra, cuando decimos "voy a comprar un pantalón vaquero azul", lo que estamos expresando son dos condiciones independientes que deben darse simultáneamente para que haga la compra, en otras palabras, mi decisión es "comprar un pantalón vaquero **Y (and)** azul", no me vale ningún pantalón que no cumpla ambas condiciones. Con la decisión de compra perfectamente formalizada, nos dirigimos a una tienda de ropa y nos muestran una serie de pantalones, cuyo análisis se recoge en la tabla siguiente:

		Valores de verdad de las entradas. Se cumplen o no las condiciones previas establecidas.		Valor de verdad de la salida. Decisión a tomar una vez analizado el cumplimiento, o no, de las condiciones previas en función del nexo lógico que las relaciona (operador lógico). En este caso **Y (and).**
		¿Es un pantalón vaquero? (Entrada A)	¿Es de color azul? (Entrada B)	Quiero un pantalón vaquero **Y (and)** azul. Decisión a tomar (Salida A·B)
Pantalón 1	Pantalón de corte clásico de color amarillo	No	No	No comprar
Pantalón 2	Pantalón de corte clásico de color azul	No	Sí	No comprar
Pantalón 3	Pantalón vaquero de color rojo	Sí	No	No comprar
Pantalón 4	Pantalón vaquero de color azul	Sí	Sí	Comprar

En las tablas de verdad deben aparecer todas las posibles combinaciones de los valores de verdad de las entradas, para obtener todos los valores posibles de la salida, tal y como ocurre en este caso. Formalizando y generalizando la tabla anterior, obtenemos la tabla de verdad del operador **Y** (**and**), tal y como se muestra en la Tabla 1.

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 1: Tabla de verdad del operador AND.

Con esta tabla ya puedo comprar con total seguridad. Para cada pantalón que me enseñen solo tengo que acudir a ella y comprobar el cumplimiento, o no, de las condiciones establecidas y tomar la decisión que corresponda, a cada caso, recogida en la última columna.

En el álgebra de proposiciones, el 1 representaría la veracidad de la proposición de que se trate y el 0 la falsedad de la misma. En la circuitería, la presencia de señal se representa por 1 y la ausencia por 0.

1.1.- Operador AND (producto).

En su forma básica, es un operador que utiliza dos entradas y genera una única salida, devolviendo el valor 1 (verdad) solo cuando ambas entradas son, simultáneamente, verdad (1) y el valor 0 (falso) en cualquier otro caso. Tal y como se muestra en la tabla de verdad mostrada en la Tabla 2.

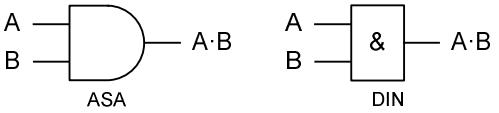


Figura 1: Representaciones de la puerta AND.

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 2: Tabla de verdad del operador AND.

En el caso de tener más de dos entradas, solo devolverá el valor 1 (verdad) cuando sean verdad (1), simultáneamente, todas las posibles entradas. Devolviendo el valor 0, en cualquier otro caso.

En los esquemas de circuitería digital, los símbolos más usados para su representación son los mostrados en la Figura 1, correspondientes a las normativas ASA y DIN. En el estándar DIN, se utiliza el símbolo & para indicar que la operación *and* se realiza dentro del bloque.

1.2.- Operador OR (suma).

En su forma básica, es un operador que utiliza dos entradas y genera una única salida, devolviendo el valor 1 (verdad) cuando alguna, o ambas, de las entradas son verdad (1) y el valor 0 (falso) solo cuando son falsas, simultáneamente, ambas entradas. Tal y como se muestra en la tabla de verdad recogida en la Tabla 3.

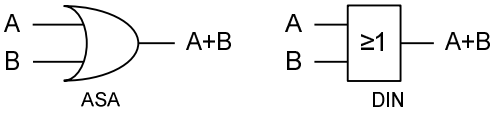


Figura 2: Representaciones de la puerta OR.

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Tabla 3: Tabla de verdad del operador OR.

En el caso de tener más de dos entradas, devolverá el valor 0 (falso) solo cuando sean falsas (0), simultáneamente, todas las posibles entradas. Devolviendo el valor 1 en cualquier otro caso.

En los esquemas de circuitería digital, los símbolos más usados para su representación son los mostrados en la Figura 2, correspondientes a las normativas ASA y DIN. En el estándar DIN, el símbolo contiene la designación ≥1 para indicar que la suma matemática, de los valores de las variables de entrada, determinan la salida de la puerta.

1.3.- Operador NOT (inversor).

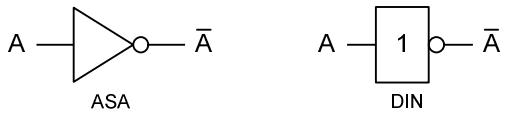


Figura 3: Representaciones de la puerta NOT.

A	$\overline{A}$
0	1
1	0

Tabla 4: Tabla de verdad del operador NOT.

Es el único operador lógico que tiene una sola entrada y una única salida (operador unario). Esta salida devuelve el valor opuesto al que recibe, razón por la cual se suele denominar *inversor*, Tabla 4.

En los esquemas de circuitería digital, los símbolos más usados para su representación son los mostrados en la Figura 3, correspondientes a las normativas ASA y DIN.

1.4.- Operadores compuestos.

Solo con los tres tipos de operadores vistos (AND, OR y NOT), se puede implementar cualquier función lógica, por compleja que sea. Sin embargo, existen otras puertas (operadores) compuestas (resultado de la agrupación de varias de las anteriores), que se utilizan profusamente en electrónica digital, y lógica, y que se representan como puertas independientes.

1.4.1.- Operador NAND (Negación de AND, producto negado).

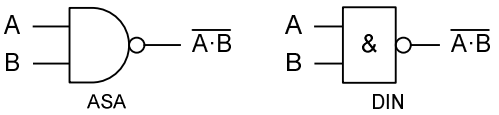


Figura 4: Representaciones de la puerta NAND.

A	B	$\overline{A·B}$
0	0	1
0	1	1
1	0	1
1	1	0

Tabla 5: Tabla de verdad del operador NAND.

Corresponde a la negación del AND, y en su forma básica, es una puerta que utiliza dos entradas y genera una única salida, devolviendo el valor 0 (falso) solo cuando ambas entradas son, simultáneamente, verdad (1) y el valor 1 (verdad) en cualquier otro caso. Tal y como se muestra en la tabla de verdad de la Tabla 5, en la que puede comprobarse que las salidas son las inversas a las obtenidas, previamente, para el operador AND, Tabla 2.

En el caso de tener más de dos entradas, solo devolverá el valor 0 (falso) cuando sean verdad (1), simultáneamente, todas las posibles entradas. Devolviendo el valor 1 en el resto de los casos.

En los esquemas de circuitería digital, los símbolos más usados para su representación son los mostrados en la Figura 4, correspondientes a las normativas ASA y DIN.

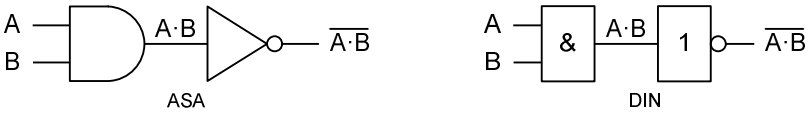


Figura 5: Composición de la puerta NAND.

Esta puerta se obtiene disponiendo una puerta AND como entrada de un inversor, tal y como se muestra en el esquema de la Figura 5, utilizando los estándares gráficos ASA y DIN.

1.4.2.- Operador NOR (Negación de OR, suma negada).

Corresponde a la negación del OR, y en su forma básica, es una puerta que utiliza dos entradas y genera una única salida, devolviendo el valor 1 (verdad) cuando ambas entradas, simultáneamente, son falsas (0) y el valor 0 (falso) en cualquier otra circunstancia. Tal y como se muestra en la tabla de verdad de la Tabla 6, en la que puede comprobarse que las salidas son las inversas a las obtenidas, previamente, para el operador OR, Tabla 3.

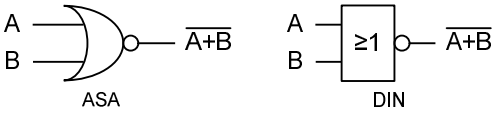


Figura 6: Representaciones de la puerta NOR.

En el caso de tener más de dos entradas, devolverá el valor 1 (verdad) solo cuando sean falsas (0), simultáneamente, todas las posibles entradas. Devolviendo el valor cero en el resto de los casos.

En los esquemas de circuitería digital, los símbolos más usados para su representación son los mostrados en la Figura 6, correspondientes a las normativas ASA y DIN.

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Tabla 6: Tabla de verdad del operador NOR.

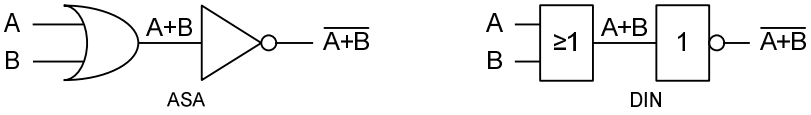


Figura 7: Composición de la puerta NOR.

Esta puerta se obtiene disponiendo una puerta OR como entrada de un inversor, tal y como se muestra en la Figura 7, utilizando los estándares gráficos ASA y DIN.

1.4.3.- Operador XOR (OR exclusivo).

En su forma básica, es una puerta que utiliza dos entradas y genera una única salida, devolviendo el valor uno (verdad) cuando alguna de las entradas, exclusivamente, es cierta (1) y el valor 0 (falso) cuando son falsas (0) o verdaderas (1), simultáneamente, ambas entradas. Tal y como se muestra en la tabla de verdad contenida en la Tabla 7.

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 7: Tabla de verdad del operador XOR.

En el caso de tener más de dos entradas, devolverá el valor 0 (falso) cuando sean falsas (0) o verdaderas (1), simultáneamente, todas las posibles entradas. Y el valor 1 en el resto de los casos.

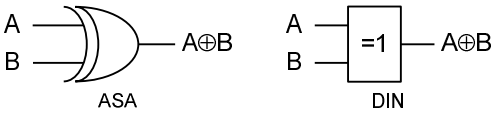
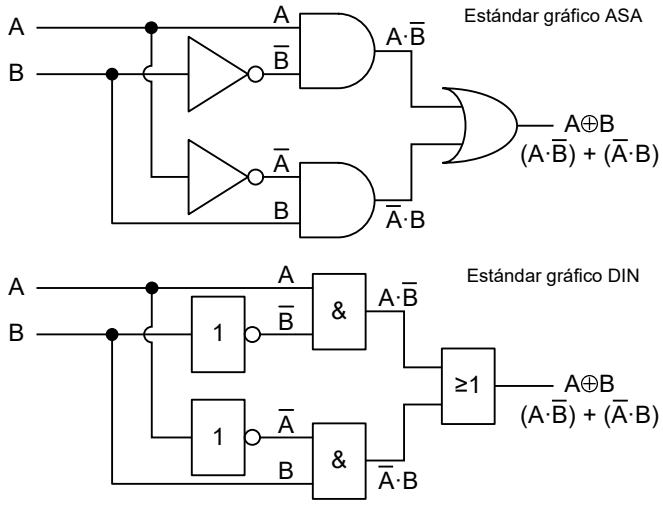


Figura 8: Representaciones de la puerta XOR.

En los esquemas de circuitería digital, los símbolos más usados para su representación son los mostrados en la Figura 8, correspondientes a las normativas ASA y DIN.

Esta puerta se obtiene de la composición de las puertas elementales que se muestra en la Figura 9, utilizando los estándares gráficos ASA y DIN, y cuya tabla de verdad se recoge en la Tabla 8. Tal y como puede comprobarse, en la tabla de verdad, el resultado final es el de un operador XOR.



A	B	$\overline{A}$	$\overline{B}$	$A\cdot\overline{B}$	$\overline{A}\cdot B$	$(A\cdot\overline{B}) + (\overline{A}\cdot B)$
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

Tabla 8: Tabla de verdad de la composición del operador XOR.

Figura 9: Composición de la puerta XOR.

2.- EJEMPLOS.

En informática, y en concreto en redes, estos operadores suelen aplicarse sobre un conjunto más o menos grande de bits (el contenido de un registro, una dirección IP, una máscara de subred, etc.) y es necesario tener en cuenta que se debe aplicar, el operador, a todos y cada uno de los bits del conjunto. Pudiendo darse dos casos, que se trate de un operador con una única entrada (inversor) o un operador de entradas múltiples.

2.1.- INVERSOR.

Cuando el operador NOT se aplica sobre un conjunto de bits, se invierten todos y cada uno de los bits del conjunto, uno a uno. Por ejemplo, el resultado de un operado NOT sobre el byte 10101010 sería 01010101, en el que se invirtió el valor de cada uno de los bits del byte.

2.2.- OPERADORES DE ENTRADAS MÚLTIPLES.

Cuando se trata de operadores que requieren más de una entrada, el operador se aplica, bit a bit, sobre los bits de las entradas que ocupan posiciones idénticas.

Por ejemplo, aplicaremos el operador AND a los siguientes bytes; 10100111 y 11001001, Figura 10. Una forma de evitar equivocaciones es disponerlos como en una suma, ya que nos permite, por un lado, comprobar que tengan el mismo número

Byte 1 10100111
Byte 2 11001001
Resultado del AND..... 10000001

Figura 10

de bits ambas entradas y por otro, nos facilita la operación evitando errores, sobre todo cuando el número de bits a tratar sea grande.

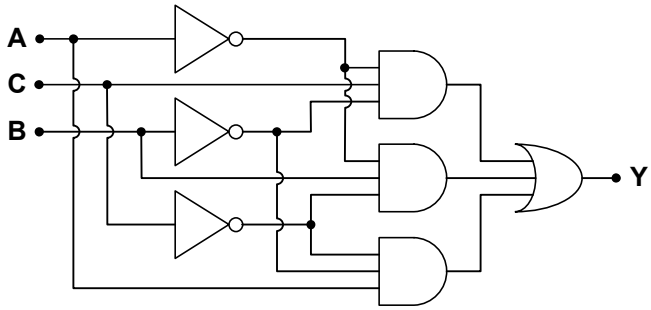
3.- EJERCICIOS PROPUESTOS.

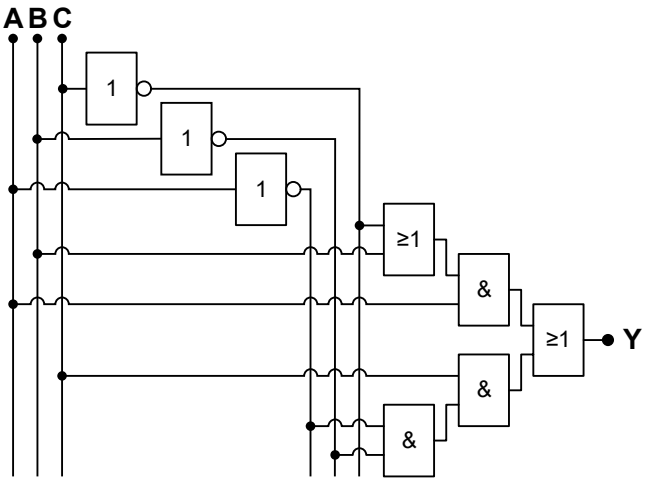
1.- Aplicar los operadores AND, NAND, OR, NOR y XOR a los siguientes conjuntos de bits: 1011000101 y 1110100110.

2.- Dada la IP 192.168.0.171, con máscara de subred 255.255.255.0, aplicar un AND entre ellos y expresar el resultado en formato decimal con puntos.

3.- Dada la IP 203.163.168.55, con máscara de subred FFF00000, aplicar un AND entre ellos y expresar el resultado en formato decimal con puntos.

4.- Obtener la tabla de verdad de los siguientes circuitos lógicos:





4.- EJERCICIOS PROPUESTOS RESUELTOS.

1.- Aplicar los operadores AND, NAND, OR, NOR y XOR a los siguientes conjuntos de bits: 1011000101 y 1110100110.

	AND	NAND	OR	NOR	XOR
Entrada 1	1011000101	1011000101	1011000101	1011000101	1011000101
Entrada 2	1110100110	1110100110	1110100110	1110100110	1110100110
Resultados	1010000100	0101111011	1111100111	0000011000	0101100011

Adviértase que en el caso del NAND se obtiene como resultado la negación del obtenido para el AND, y viceversa, ocurriendo lo mismo en los resultados alcanzados para el operador NOR y el nexo OR, correspondiendo el uno a la negación del otro.

2.- Dada la IP 192.168.0.171, con máscara de subred 255.255.255.0, aplicar un AND entre ellos y expresar el resultado en formato decimal con puntos.

	Formato decimal con puntos (representación decimal de cada uno de los cuatro bytes que las componen, separados por puntos)	En binario (representación binaria de cada uno de los cuatro bytes que se representaban en base decimal y sin puntos)
IP	192.168.0.171	11000000 10101000 00000000 10101011
Máscara de subred	255.255.255.0	11111111 11111111 11111111 00000000
	Resultado del AND en binario	11000000 10101000 00000000 00000000
	Resultado del AND en formato decimal con puntos	192.168.0.0

Obsérvese que la representación binaria del 0 decimal (192.168.0.171) debe normalizarse a los ocho bits que posee el byte que se muestra en el formato decimal con puntos.

El resultado obtenido es lo que se conoce como dirección de red o genérico de la red (hace referencia a la red en su conjunto, no puede usarse como IP de un equipo), y nos indica que la IP dada pertenece a la subred 192.168.0.0, en concreto se trataría del equipo (*host*) 171 de la subred 192.168.0.0, con máscara 255.255.255.0. Otra forma, más breve, de indicar la IP y la máscara correspondiente, es 192.168.0.171/24, en donde se expresa la IP, en formato decimal con puntos, y el número de bits a uno de la máscara (en este caso, 24), separados por una barra inclinada (/). A esta forma de representación se le suele llamar "formato de barra inclinada" o, más técnicamente, "notación CIDR", siendo CIDR[4] las siglas de ***C**lassless **I**nter-**D**omain **R**outing*, que podría traducirse como "enrutamiento entre dominios sin clases"; cuyo significado comprenderá en los próximos días.

Esta es una de las primeras operaciones que realiza un ordenador cuando arranca, para saber a qué subred pertenece y construir la llamada tabla de enrutado, que le servirá como guía de viaje de cómo llegar a su propia red, o a cualquier otra.

En un sistema *Windows*, podemos ver la tabla de enrutado de nuestro equipo mediante la ejecución del comando ***route print***, que mostrará algo similar a lo que puede verse en la Figura 11, en la que se señaló la entrada generada mediante la operación AND entre la propia IP y la máscara de subred correspondiente. En términos sencillos, esta entrada de la tabla le indica, al equipo, que cuando quiera comunicarse con cualquier otro nodo de esa subred (su propia subred) tan "solo" tiene que indicar la IP del equipo destino y soltar la trama, por su interfaz de red, a la red. El equipo destino la recibirá directamente.

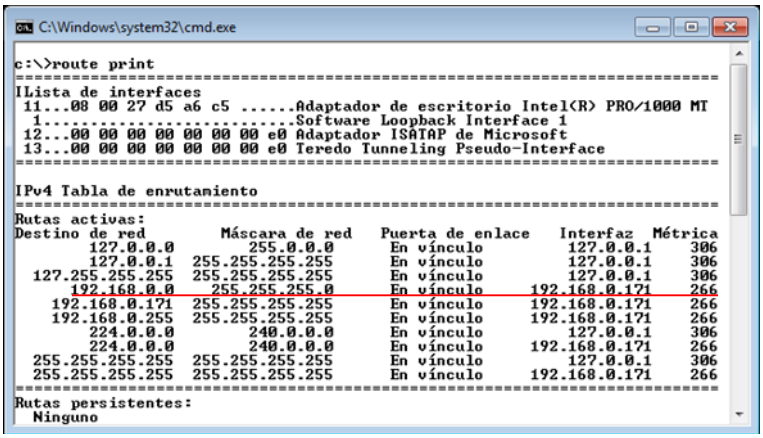


Figura 11

Resultado equivalente se obtiene, en sistemas *GNU/Linux*, ejecutando el comando ***ip route***, tal y como se muestra en la Figura 12.

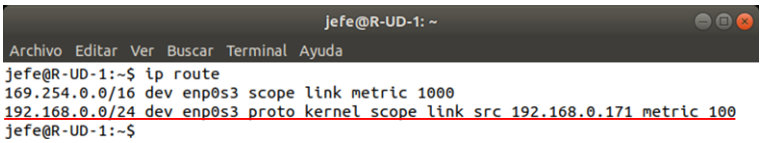


Figura 12

Adviértase, que gracias al operador AND, y a su tabla de verdad, cuando nos interese conocer el valor de una serie de bits determinada, lo único que tenemos que hacer es construir una plantilla, en este caso la máscara de subred, en la cual se encuentren a 1 los bits de las posiciones cuyo valor nos interese conocer, y el resto de los bits de la plantilla los dejamos a cero. El resultado será una copia del valor de los bits que nos importan y el resto a ceros.

3.- Dada la IP 203.163.168.55, con máscara de subred FFF00000, aplicar un AND entre ellos y expresar el resultado en formato decimal con puntos.

	En los formatos dados	En binario
IP	203.163.168.55	11001011 10100011 10101000 00110111
Máscara de subred	FFF00000	11111111 11110000 00000000 00000000

Resultado del AND en binario	11001011 10100000 00000000 00000000
Resultado del AND en formato decimal con puntos	203.160.0.0

Obsérvese que la representación binaria del 55 decimal (110111) es necesario normalizarla a 8 bits al convertir del formato decimal con puntos, a la codificación binaria, añadiendo 2 ceros a la izquierda (**00**110111).

En este caso, la obtención de la dirección de red no es tan evidente como en el ejercicio anterior, ya que no coincide con los bytes correspondientes de la IP. Eso se debe, a que la máscara de subred, en formato decimal con puntos, es 255.240.0.0 lo que nos indica que tan solo se utilizan 12 bits (8+4, correspondientes a los bytes de mayor peso, en hexadecimal corresponden a los dígitos FFF) como plantilla para identificar la red.

Según esto, sería el nodo 3.168.55 de la subred 203.160.0.0, con máscara 255.240.0.0, que habitualmente se expresaría como 203.163.168.55/12, en el formato CIDR.

En este supuesto, la entrada que se generaría, en la correspondiente tabla de enrutado, sería la mostrada en la Figura 13, en el caso de sistemas *Windows* o *GNU/Linux*.

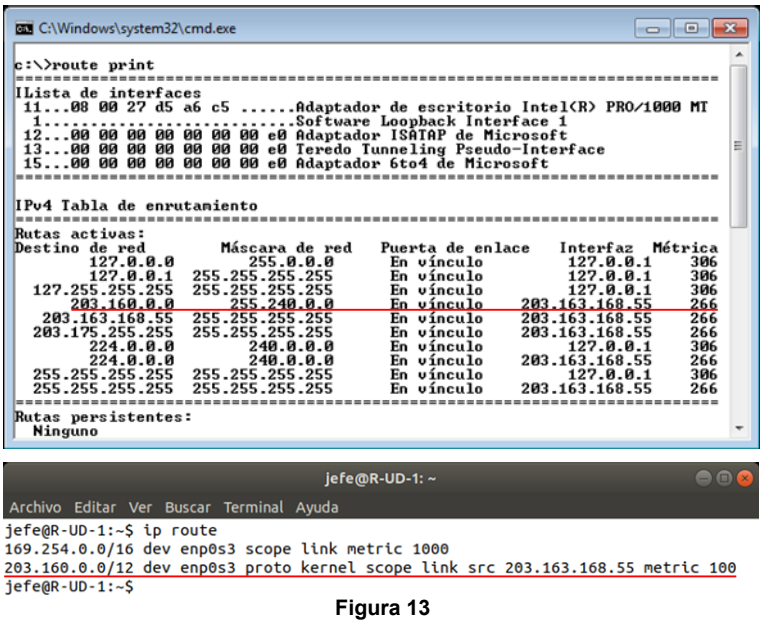
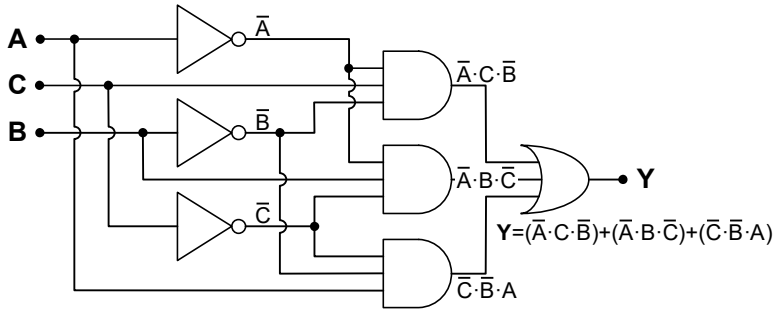


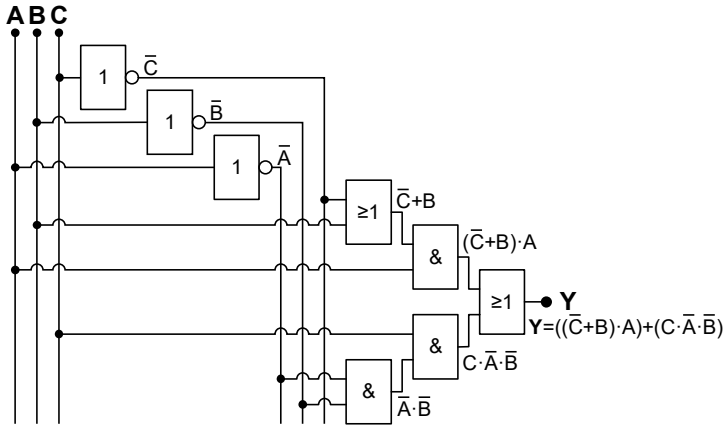
Figura 13

4.- Obtener la tabla de verdad de los siguientes circuitos lógicos:

Para obtener la tabla de verdad pedida, en cada caso, empezaremos por determinar que salida se obtiene en cada una de las puertas, empezando de izquierda a derecha, y añadiendo en la tabla la columna correspondiente a las distintas salidas de la puerta analizada, hasta llegar al final del circuito. Tal y como se muestra en las figuras siguientes.



A	B	C	$\bar{A}$	$\bar{B}$	$\bar{C}$	$\bar{A}\cdot C\cdot\bar{B}$	$\bar{A}\cdot B\cdot\bar{C}$	$\bar{C}\cdot\bar{B}\cdot A$	Y
0	0	0	1	1	1	0	0	0	0
0	0	1	1	1	0	1	0	0	1
0	1	0	1	0	1	0	1	0	1
0	1	1	1	0	0	0	0	0	0
1	0	0	0	1	1	0	0	1	1
1	0	1	0	1	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0	0	0



A	B	C	$\bar{A}$	$\bar{B}$	$\bar{C}$	$\bar{C}+B$	$\bar{A}\cdot\bar{B}$	$(\bar{C}+B)\cdot A$	$C\cdot\bar{A}\cdot\bar{B}$	Y
0	0	0	1	1	1	1	1	0	0	0
0	0	1	1	1	0	0	1	0	1	1
0	1	0	1	0	1	1	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0
1	0	0	0	1	1	1	0	1	0	1
1	0	1	0	1	0	0	0	0	0	0
1	1	0	0	0	1	1	0	1	0	1
1	1	1	0	0	0	1	0	1	0	1

5.- NOTAS FINALES.

[1] En la actualidad se denomina **ANSI** (***A**merican **N**ational **S**tandards **I**nstitute*, Instituto Nacional Estadounidense de Estándares). Véase: https://www.ansi.org/

[2] Para DIN véase: https://www.din.de/en

[3] Para álgebra de proposiciones véase: https://es.wikipedia.org/wiki/L%C3%B3gica_proposicional

[4] Para CIDR véase: https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing