

### Representación en Punto fijo: Números sin y con signo. Operaciones aritméticas. Flags

**Objetivos de la práctica:** que el alumno sea capaz de:

- Representar e interpretar números sin signo y con signo.
- Realizar operaciones aritméticas e interpretar los flags de acarreo, cero, overflow y negativo.

**Bibliografía:**

- “Organización y Arquitectura de Computadoras” de W. Stallings, capítulo 8.
- Apunte 1 de la cátedra, “Sistemas de Numeración: Sistemas Enteros y Punto Fijo”.

1. Represente los siguientes números en el sistema BSS y en en los sistemas BCS, Ca1, Ca2 y Ex2, todos restringidos a 8 bits. En los casos que no se pueda representar, aclarar por qué.

**0; 1; 127; 128; 255; 256; -1; -7; -127; -128; -139; -56; 130; 45; 90; -90; 0,75; 2,5.**

Recuerde: Los positivos se representan igual en los sistemas BSS, BCS, Ca1 y Ca2 (ver representación de números en binario en el apunte). Los negativos en BCS, signo en el bit de mayor peso (0 positivos y 1 negativos) y los restantes son módulo. Los negativos en Ca1, se obtiene el BSS del número en 8 bits, y luego se cambian unos por ceros y ceros por unos. Los negativos en Ca2, se obtienen sumando 1 a la representación de Ca1, o copiando hasta el primer 1 (incluido) desde la derecha el número en BSS, y luego se cambian unos por ceros y ceros por unos. En Ex2, se suma siempre el exceso (que en n bits será  $2^{n-1}$ ) y luego se representa como BSS.

2. Interprete las siguientes cadenas de 8 bits en los sistemas BSS, BCS, Ca1, Ca2 y Ex2.

**00000000; 00000001; 11111110; 01111111; 11111111; 00010001; 10011001; 10101010; 01100110;**

3. Calcule el rango y resolución de un sistema de punto fijo en BSS con 7 bits de parte entera y 3 de fraccionaria y de un sistema de punto fijo en BCS con 1 bit de signo, 5 bits de parte entera y 4 de fraccionaria.

4. Represente los siguientes números en los sistemas del ejercicio 3. Si no es posible obtener una representación exacta, indique cuál es la más próxima y calcule en ese caso el error cometido. Si el número a representar está fuera del rango del sistema, señale que ese número “NO SE PUEDE REPRESENTAR”.

**7 ; 15,125 ; 2,2 ; 8,001 ; 123,25 ; 50,50 ; 120 ; 1,2 ; 1,25 ; 35 ; -1,25 ; 1,0625 ; -1,5625 ; -35,5**

5. Interprete las siguientes cadenas en los sistemas del ejercicio 3.

**0000000000 ; 0101010101 ; 1000000000 ; 1111111110 ; 1111111111 ; 1010101010 ; 0111111111 ; 0110110110**

6. Represente los números **0, 1, 3, 8, 12, 13, 22, 35, 99, 100 y 1255** en los sistemas BCD y BCD empaquetado. Describa, con el mayor nivel de detalle posible, un procedimiento para calcular sumas en BCD. Sin considerar representación de signo, realice las siguientes operaciones en BCD: **32 + 45; 22 + 89; 1307 + 708**

7. Escriba los números **13160, 2988, 927 y 87127** en los sistemas BCD, BCD empaquetado y BSS. Observe la cantidad de bits necesarios. ¿Qué conclusiones saca respecto de las ventajas y desventajas del sistema BCD sobre BSS?

8. Haga el pasaje de binario a hexadecimal y de hexadecimal a BCH en forma directa (sin utilizar sistema decimal). ¿Por qué cree que el sistema hexadecimal es muy utilizado?

| Binario a Hexadecimal |  |
|-----------------------|--|
| 1010000010000         |  |
| 1110001011101         |  |
| 111010011001011       |  |
| 1001111100100011      |  |
| 1110101011001010      |  |
| 101101101011010       |  |

| Hexadecimal a BCH |  |
|-------------------|--|
| 2801              |  |
| 1C5D              |  |
| 78AB              |  |
| F79A              |  |
| 7EF1              |  |
| 324A              |  |

## Organización de Computadoras 2023

9. Calcule el resultado de realizar las sumas (ADD) y restas (SUB) indicadas a continuación. Calcule el valor en el que quedarán los flags luego de realizada cada operación, de acuerdo a que haya habido acarreo (flag C, de Carry) o se haya producido borrow (flag B, es el mismo que C pero en la resta), o que el resultado sea cero en todos sus bits (flag Z, de Zero), se haya producido desbordamiento (flag V, de overflow), o de un resultado negativo (flag N, de Negative).

|              |          |              |          |              |          |              |          |
|--------------|----------|--------------|----------|--------------|----------|--------------|----------|
| ADD 00011101 | 00011011 | ADD 01110000 | 11110001 | SUB 00011101 | 00011011 | SUB 01110000 | 11110001 |
| ADD 10011101 | 01110010 | ADD 01001100 | 01110000 | SUB 10011101 | 01110010 | SUB 01001100 | 01110000 |
| ADD 01110110 | 01110001 | ADD 11001100 | 11110000 | SUB 01110110 | 01110001 | SUB 11001100 | 11110000 |
| ADD 10111001 | 11100011 | ADD 10000000 | 10000000 | SUB 10111001 | 11100011 | SUB 10000000 | 10000000 |
| ADD 00111010 | 00001111 | ADD 00000000 | 10000000 | SUB 00111010 | 00001111 | SUB 00000000 | 10000000 |

Recuerde que:

|               |               |               |               |
|---------------|---------------|---------------|---------------|
| 0+0=0 con C=0 | 1+0=1 con C=0 | 0-0=0 con B=0 | 1-1=0 con B=0 |
| 0+1=1 con C=0 | 1+1=0 con C=1 | 1-0=1 con B=0 | 0-1=1 con B=1 |

También, tendremos casos de exceso en el rango de representación (llamado overflow) si a un número positivo se le suma otro positivo y da un resultado negativo ó a un número negativo se le suma otro negativo y da uno positivo ó a un número positivo se le resta otro negativo y da uno negativo ó a un número negativo se le resta otro positivo y da uno positivo.

En todos estos casos de errores en la operación aritmética, se advierte el error pues la ALU encenderá (pondrá en 1) el flag de overflow (V=1). Es de hacer notar que el flag V se encenderá aunque sumemos números sin signo (en BSS), la interpretación de los flags corre por cuenta del programador.

Ejemplo de suma:

|            |            |
|------------|------------|
| 1111       |            |
| 01001111   | □ Acarreos |
| + 01111000 |            |
| 11000111   |            |

Flags: Carry=0; Zero=0; Negative=1; overflow=1.

10. Suponga que los operandos del ejercicio anterior (ej. 9) eran números representados en BSS, BCS, Ca1, Ca2 y Exceso2 (todos para cada sistema de representación). Verifique la correctitud del resultado interpretando el resultado obtenido y comparando con el resultado esperado. En caso de que la operación haya dado resultado incorrecto, indicar la posible cadena de bits que representa el resultado correcto.

Del ejemplo anterior, los operandos y resultado son interpretados como cadenas de bits BSS.

|            |            |                       |
|------------|------------|-----------------------|
| 1111       |            | Interpretación en BSS |
| 01001111   | □ Acarreos |                       |
| + 01111000 |            |                       |
| 11000111   |            |                       |

|       |
|-------|
| 79    |
| + 120 |
| 199   |

Por lo que, si verificamos realizando a mano la operación interpretada en base 10, el resultado es correcto.

Volviendo al ejemplo, si interpretamos ahora los operandos y resultados como cadenas de bits en los 4 sistemas de representación de números con signo, tendremos:

|            |            |     |     |     |        |                    |
|------------|------------|-----|-----|-----|--------|--------------------|
| 1111       |            | BCS | Ca1 | Ca2 | Exceso | Observe los flags! |
| 01001111   | □ Acarreos |     |     |     |        |                    |
| + 01111000 |            |     |     |     |        |                    |
| 11000111   |            |     |     |     |        |                    |

|       |       |       |      |
|-------|-------|-------|------|
| 79    | 79    | 79    | -49  |
| + 120 | + 120 | + 120 | + -8 |
| -71   | -56   | -57   | 71   |

11. Referido al ejercicio 9 sobre la operación ADD: Observando cuáles resultados fueron correctos y cuáles fueron incorrectos y relacionándolos con los flags, describa una regla para determinar la correctitud de la operación ADD en el sistema BSS con la mera observación de los flags (sin verificar la operación pasando por el sistema decimal). Observe que en el ejemplo dado para BSS, los flags V y N quedan en 1 y no importan pues suponemos que estamos operando con números sin signo (BSS). Si hacemos lo mismo con todos los ejercicios, observaremos que en los casos en que C=1 el resultado es incorrecto, independientemente de los demás flags.
12. Trabaje de forma similar al ejercicio 10 pero con la operación SUB. Luego trate de descubrir reglas análogas para ADD y SUB para el sistema Ca2, basándose en los ejercicios cuya cadena resultado es diferente de la correcta y observando los flags. Observe qué flags se encienden en los casos que da incorrecto y cuáles no, como así también los que es indistinto que tengan valor uno o cero.
13. Considere en el ejercicio 9, que el punto o coma fraccionaria se encuentra entre el bit 2 y el 3. Interprete el valor que tendrán las cadenas de bits que representan los operandos y los resultados como BSS y como Ca2. Observe los flags. ¿Qué concluye?

## Organización de Computadoras 2023

14. Escriba todas las cadenas de los sistemas BSS, BCS, Ca1, Ca2 y  $\text{Ex}2^{(n-1)}$  restringido a 4 bits. Considere el punto (o coma fraccionaria) fijo en cada una de todas las posibles posiciones (son 5 posibilidades en total, considerando que el punto fijo puede estar colocado a la izquierda del MSB y a la derecha del LSB) y obtenga el rango y resolución de cada uno de los sistemas de punto fijo resultantes. ¿Cuántas cadenas se pueden escribir en cada caso? ¿Cuántos números se pueden representar en los distintos sistemas?
15. Defina el sistema Exceso a M (donde M es un entero cualquiera).
16. Describa mecanismos para sumar y restar en BCS, Ca1 y Exceso, en base al análisis de los resultados y flags del punto 9, realizando la interpretación de los operandos y resultados en los distintos sistemas de representación citados. Observe de qué manera (qué operaciones deberían realizarse y en qué caso) se llegaría al resultado correcto.
17. Interprete las siguientes cadenas descritas en sistema Ca2. ¿Qué pasa en el caso (e)?  
**a. 00100110      b. 11011000      c. 00111000      d. 00000000      e. 10000000**
18. Interprete las siguientes cadenas descritas en sistema  $\text{Ex}2^{(n-1)}$  con  $n=8$ . ¿Qué pasa en el caso (e)?  
**a. 10100110      b. 01011000      c. 10111000      d. 10000000      e. 00000000**

# Organización de Computadoras

## Práctica 1 Representación en Punto Fijo

*Números sin signo y números con signo. Operaciones aritméticas. Flags.*

### OBJETIVOS

1. Representar e interpretar números sin signo y números con signo.
2. Realizar operaciones aritméticas e interpretar los flags de acarreo, cero, overflow y negativo.

### BIBLIOGRAFÍA

1. “Organización y Arquitectura de Computadoras” de W. Stallings, capítulo 8.
2. Apunte 1 de la cátedra, “Sistemas de Numeración: Sistemas Enteros y Punto Fijo”.

---

**1. Represente cada uno de los siguientes números en los sistema BSS, BCS, Ca1, Ca2 y Ex2, todos restringidos a 8 bits. En los casos que no se pueda representar, aclarar por qué.**

Los positivos se representan igual en los sistemas BSS, BCS, Ca1 y Ca2. Para estas representaciones y para Ex2 y números negativos, consultar el apunte y material adicional sobre números binarios.

| Decimal | BSS | BCS | Ca1 | Ca2 | Ex2 |
|---------|-----|-----|-----|-----|-----|
| 0       |     |     |     |     |     |
| 1       |     |     |     |     |     |
| 45      |     |     |     |     |     |
| 90      |     |     |     |     |     |
| 127     |     |     |     |     |     |
| 128     |     |     |     |     |     |
| 130     |     |     |     |     |     |
| 255     |     |     |     |     |     |

|      |  |  |  |  |  |
|------|--|--|--|--|--|
| 256  |  |  |  |  |  |
| -1   |  |  |  |  |  |
| -7   |  |  |  |  |  |
| -56  |  |  |  |  |  |
| -90  |  |  |  |  |  |
| -127 |  |  |  |  |  |
| -128 |  |  |  |  |  |
| -139 |  |  |  |  |  |
| 0,75 |  |  |  |  |  |
| 2,5  |  |  |  |  |  |

## 2. Interprete las siguientes cadenas de 8 bits en los sistemas BSS, BCS, Ca1, Ca2 y Ex2.

| Cadena   | BSS | BCS | Ca1 | Ca2 | Ex2 |
|----------|-----|-----|-----|-----|-----|
| 00000000 |     |     |     |     |     |
| 00000001 |     |     |     |     |     |
| 11111110 |     |     |     |     |     |
| 01111111 |     |     |     |     |     |
| 11111111 |     |     |     |     |     |
| 00010001 |     |     |     |     |     |
| 10101010 |     |     |     |     |     |
| 01100110 |     |     |     |     |     |

3. Calcule el rango y resolución de un sistema de punto fijo en BSS con 7 bits de parte entera y 3 de fraccionaria y de un sistema de punto fijo en BCS con 1 bit de signo, 5 bits de parte entera y 4 de fraccionaria.

| Sistema   | Rango          |                | Resolución |
|---|----------------|----------------|------------|
|   | Desde (mínimo) | Hasta (máximo) |            |
| BSS con 7 bits de parte entera y 3 bits de parte fraccionaria                 |                |                |            |
| BCS con 1 bit de signo, 5 bits de parte entera y 4 bits de parte fraccionaria |                |                |            |

4. Represente los siguientes números en los sistemas del ejercicio 3. Si no es posible obtener una representación exacta, indique cuál es la más próxima y calcule en ese caso el error cometido. Si el número a representar está fuera del rango del sistema, señale que ese número "NO SE PUEDE REPRESENTAR".

| Decimal | BSS con 7 bits de parte entera y 3 bits de parte fraccionaria | BCS con 1 bit de signo, 5 bits de parte entera y 4 bits de parte fraccionaria |
|---------|---|---|
| 7       |   |   |
| 15,125  |   |   |
| 2,2     |   |   |
| 8,001   |   |   |
| 123,25  |   |   |
| 50,50   |   |   |
| 120     |   |   |

|         |  |  |
|---------|--|--|
| 1,2     |  |  |
| 1,25    |  |  |
| 35      |  |  |
| -1,25   |  |  |
| 1,0625  |  |  |
| -1,5625 |  |  |
| -35,5   |  |  |

5. Interprete las siguientes cadenas en los sistemas del ejercicio 3.

| Cadena     | BSS con 7 bits de parte entera y 3 bits de parte fraccionaria | BCS con 1 bit de signo, 5 bits de parte entera y 4 bits de parte fraccionaria |
|------------|---|---|
| 0000000000 |   |   |
| 0101010101 |   |   |
| 1000000000 |   |   |
| 1111111110 |   |   |
| 1111111111 |   |   |
| 1010101010 |   |   |

|            |  |  |
|------------|--|--|
| 0111111111 |  |  |
| 0110110110 |  |  |

6a. Represente los números 0, 1, 3, 8, 12, 13, 22, 35, 99, 100 y 1255 en los sistemas BCD y BCD empaquetado.

| Decimal | BCD | BCD empaquetado |
|---------|-----|-----------------|
| 0       |     |                 |
| 1       |     |                 |
| 3       |     |                 |
| 8       |     |                 |
| 12      |     |                 |
| 13      |     |                 |
| 22      |     |                 |
| 35      |     |                 |
| 99      |     |                 |
| 100     |     |                 |
| 1255    |     |                 |



6b. Describa, con el mayor nivel de detalle posible, un procedimiento para calcular sumas en BCD.

6c. Sin considerar representación de signo, realice las siguientes operaciones en BCD

| Operación  | Reescribir la operación en BCD | Resultado en BCD |
|------------|--------------------------------|------------------|
| 32 + 45    |                                |                  |
| 22 + 89    |                                |                  |
| 1307 + 708 |                                |                  |

7. Escriba los números 13160, 2988, 927 y 87127 en los sistemas BCD, BCD empaquetado y BSS. Observe la cantidad de bits necesarios. ¿Qué conclusiones saca respecto de las ventajas y desventajas del sistema BCD sobre BSS?

| Decimal | BCD | BCD empaquetado | BSS |
|---------|-----|-----------------|-----|
| 13160   |     |                 |     |
| 2988    |     |                 |     |
| 927     |     |                 |     |
| 87127   |     |                 |     |

8. Haga el pasaje de binario a hexadecimal y de hexadecimal a BCH en forma directa (sin utilizar sistema decimal). ¿Por qué cree que el sistema hexadecimal es muy utilizado?

| Binario a Hexadecimal |  | Hexadecimal a BCH |  |
|-----------------------|--|-------------------|--|
| 1010000010000         |  | 2801              |  |
| 1110001011101         |  | 1C5D              |  |
| 111010011001011       |  | 78AB              |  |
| 1001111100100011      |  | F79A              |  |
| 1110101011001010      |  | 7EF1              |  |
| 101101101011010       |  | 324A              |  |

9. Calcule el resultado de realizar las sumas (ADD) y restas (SUB) indicadas a continuación. Calcule el valor en el que quedarán los flags luego de realizada cada operación. La cantidad de bits de los operandos restringe la cantidad de bits del resultado.

El flag C indica acarreo en la suma o borrow en la resta.  
 El flag Z indica que todos los bits del resultado son cero.  
 El flag V indica que se produjo overflow si se interpreta en Ca2.  
 El flag N indica un resultado negativo si se interpreta en Ca2.

| Operación             | Resultado | C | Z | V | N |
|-----------------------|-----------|---|---|---|---|
| 00011101<br>+00011011 |           |   |   |   |   |
| 01110000<br>+11110001 |           |   |   |   |   |
| 00011101<br>-00011011 |           |   |   |   |   |
| 01110000<br>-11110001 |           |   |   |   |   |
| 10011101<br>+01110010 |           |   |   |   |   |

|                       |  |  |  |  |  |
|-----------------------|--|--|--|--|--|
| 01001100<br>+01110000 |  |  |  |  |  |
| 10011101<br>-01110010 |  |  |  |  |  |
| 01001100<br>-01110000 |  |  |  |  |  |
| 01110110<br>+01110001 |  |  |  |  |  |
| 11001100<br>+11110000 |  |  |  |  |  |
| 01110110<br>-01110001 |  |  |  |  |  |
| 11001100<br>-11110000 |  |  |  |  |  |
| 10111001<br>+11100011 |  |  |  |  |  |
| 10000000<br>+10000000 |  |  |  |  |  |
| 10111001<br>-11100011 |  |  |  |  |  |
| 10000000<br>-10000000 |  |  |  |  |  |
| 00111010<br>+00001111 |  |  |  |  |  |
| 00000000<br>+10000000 |  |  |  |  |  |

|                       |  |  |  |  |  |
|-----------------------|--|--|--|--|--|
| 00111010<br>-00001111 |  |  |  |  |  |
| 00000000<br>-10000000 |  |  |  |  |  |

10. Suponga que las cadenas de cada operación del ejercicio 9 eran números representados en BSS, BCS, Ca1, Ca2 y Exceso2. Interprete el resultado obtenido y verifique si fue correcto. En caso de que la operación haya dado resultado incorrecto, indicar la posible cadena de bits que representa el resultado correcto.

| Operación             | ¿Fue correcta si se interpreta en... ? |     |     |     |     |
|-----------------------|--|-----|-----|-----|-----|
|                       | BSS                                    | BCS | Ca1 | Ca2 | Ex2 |
| 00011101<br>+00011011 |  |     |     |     |     |
| 01110000<br>+11110001 |  |     |     |     |     |
| 00011101<br>-00011011 |  |     |     |     |     |
| 01110000<br>-11110001 |  |     |     |     |     |
| 10011101<br>+01110010 |  |     |     |     |     |
| 01001100<br>+01110000 |  |     |     |     |     |
| 10011101<br>-01110010 |  |     |     |     |     |

|                       |  |  |  |  |  |
|-----------------------|--|--|--|--|--|
| 01001100<br>-01110000 |  |  |  |  |  |
| 01110110<br>+01110001 |  |  |  |  |  |
| 11001100<br>+11110000 |  |  |  |  |  |
| 01110110<br>-01110001 |  |  |  |  |  |
| 11001100<br>-11110000 |  |  |  |  |  |
| 10111001<br>+11100011 |  |  |  |  |  |
| 10000000<br>+10000000 |  |  |  |  |  |
| 10111001<br>-11100011 |  |  |  |  |  |
| 10000000<br>-10000000 |  |  |  |  |  |
| 00111010<br>+00001111 |  |  |  |  |  |
| 00000000<br>+10000000 |  |  |  |  |  |
| 00111010<br>-00001111 |  |  |  |  |  |
| 00000000<br>-10000000 |  |  |  |  |  |

11. Referido al ejercicio 9 sobre la operación ADD: Observando cuáles resultados fueron correctos y cuáles fueron incorrectos y relacionándolos con los flags, describa una regla para determinar la correctitud de la operación ADD en el sistema BSS con la mera observación de los flags (sin verificar la operación pasando de binario a decimal).

---

12. Trabaje de forma similar al ejercicio 11 pero con la operación SUB en el sistema BSS. Luego trate de descubrir reglas análogas para ADD y SUB para el sistema Ca2, basándose en los ejercicios cuya cadena resultado es diferente de la correcta y observando los flags.

---

13. Considere en el ejercicio 9, que el punto o coma fraccionaria se encuentra entre el bit 2 y el 3. Interprete el valor que tendrán las cadenas de bits que representan los operandos y los resultados como BSS y como Ca2. Observe los flags. ¿Qué concluye con respecto a la correctitud de los resultados?

Por ejemplo, considere a la primera operación solicitada como se muestra a la derecha.

000111,01

Recuerde que la coma es solo ilustrativa y no forma parte de las cadenas binarias.

+000110,11

---

14. Escriba todas las cadenas de los sistemas BSS, BCS, Ca1, Ca2 y  $Ex2^{n-1}$  restringido a 4 bits. Considere el punto (o coma fraccionaria) fijo en cada una de todas las posibles posiciones (son 5 posibilidades en total, considerando que el punto fijo puede estar colocado a la izquierda del MSB y a la derecha del LSB) y obtenga el rango y resolución de cada uno de los sistemas de punto fijo resultantes. ¿Cuántas cadenas se pueden escribir en cada caso? ¿Cuántos números se pueden representar en los distintos sistemas?

---

15. Defina el sistema Exceso a M (donde M es un entero cualquiera).

---

16. Describa mecanismos para sumar y restar en BCS, Ca1 y Exceso, en base al análisis de los resultados y flags del punto 9, realizando la interpretación de los operandos y resultados en los distintos sistemas de representación citados. Observe de qué manera (qué operaciones deberían realizarse y en qué caso) se llegaría al resultado correcto.

17. Interprete las siguientes cadenas en los sistemas  $\text{Ca}_2$  y  $\text{Ex}2^{n-1}$  (con  $n=8$ ). ¿Qué pasa en el último caso?

| Binario  | Decimal (interpretado en $\text{Ca}_2$ ) | Decimal (interpretado en $\text{Ex}2^{n-1}$ ) |
|----------|--|---|
| 00100110 |  |   |
| 11011000 |  |   |
| 00111000 |  |   |
| 00000000 |  |   |
| 10000000 |  |   |

## **Trabajo Práctico N° 1:** **Representación en Punto Fijo. Números sin Signo y Números con Signo. Operaciones Aritméticas. Flags.**

### **Ejercicio 1.**

Representar los siguientes números en el sistema BSS y en los sistemas BCS, Ca1, Ca2 y Ex2, todos restringidos a 8 bits. En los casos que no se pueda representar, aclarar por qué.

Recordar: Los positivos se representan igual en los sistemas BSS, BCS, Ca1 y Ca2 (ver representación de números en binario en el apunte). Los negativos en BCS, signo en el bit de mayor peso (0 positivos y 1 negativos) y los restantes son módulo. En los negativos en Ca1, se obtiene el BSS del número en 8 bits y, luego, se cambian unos por ceros y ceros por unos. Los negativos en Ca2 se obtienen sumando 1 a la representación de Ca1 o copiando hasta el primer 1 (incluido) desde la derecha el número en BSS y, luego, se cambian unos por ceros y ceros por unos. En Ex2, se suma siempre el exceso (que en  $n$  bits será  $2^{n-1}$ ) y, luego, se representa como BSS.

| Decimal | BSS       | BCS                 | Ca1                 | Ca2       | Ex2       |
|---------|-----------|---------------------|---------------------|-----------|-----------|
| 0       | 00000000  | 00000000 - 10000000 | 00000000 - 11111111 | 00000000  | 10000000  |
| 1       | 00000001  | 00000001            | 00000001            | 00000001  | 10000001  |
| 45      | 00101101  | 00101101            | 00101101            | 00101101  | 10101101  |
| 90      | 01011010  | 01011010            | 01011010            | 01011010  | 11011010  |
| 127     | 01111111  | 01111111            | 01111111            | 01111111  | 11111111  |
| 128     | 10000000  | ---                 | ---                 | ---       | ---       |
| 130     | 10000010  | ---                 | ---                 | ---       | ---       |
| 255     | 11111111  | ---                 | ---                 | ---       | ---       |
| 256     | ---       | ---                 | ---                 | ---       | ---       |
| -1      | ---       | 10000001            | 11111110            | 11111111  | 01111111  |
| -7      | ---       | 10000111            | 11111000            | 11111001  | 01111001  |
| -56     | ---       | 10111000            | 11000111            | 11001000  | 01001000  |
| -90     | ---       | 11011010            | 10100101            | 10100110  | 00100110  |
| -127    | ---       | 11111111            | 10000000            | 10000001  | 00000001  |
| -128    | ---       | ---                 | ---                 | 10000000  | 00000000  |
| -139    | ---       | ---                 | ---                 | ---       | ---       |
| 0,75    | 000000,11 | 000000,11           | 000000,11           | 000000,11 | 10000,11  |
| 2,5     | 0000010,1 | 0000010,1           | 0000010,1           | 0000010,1 | 1000010,1 |

En los casos que no se puede representar, es debido al rango de representación de los diferentes sistemas:

BSS:  $0 \leq X \leq 2^n - 1 = [0, 255]$ .

BCS:  $-(2^{n-1} - 1) \leq X \leq 2^{n-1} - 1 = [-127, 127]$ .

Ca1:  $-(2^{n-1} - 1) \leq X \leq 2^{n-1} - 1 = [-127, 127]$ .

Ca2:  $-2^{n-1} \leq X \leq 2^{n-1} - 1 = [-128, 127]$ .

Ex2:  $-2^{n-1} \leq X \leq 2^{n-1} - 1 = [-128, 127]$ .



**Ejercicio 2.**

*Interpretar las siguientes cadenas de 8 bits en los sistemas BSS, BCS, Ca1, Ca2 y Ex2.*

| <b>Cadena</b> | <b>BSS</b> | <b>BCS</b> | <b>Ca1</b> | <b>Ca2</b> | <b>Ex2</b> |
|---------------|------------|------------|------------|------------|------------|
| 00000000      | 0          | 0          | 0          | 0          | -128       |
| 00000001      | 1          | 1          | 1          | 1          | -127       |
| 11111110      | 254        | -126       | -1         | -2         | 126        |
| 01111111      | 127        | 127        | 127        | 127        | -1         |
| 11111111      | 255        | -127       | 0          | -1         | 127        |
| 00010001      | 17         | 17         | 17         | 17         | -111       |
| 10011001      | 153        | -25        | -102       | -103       | 25         |
| 10101010      | 170        | -42        | -85        | -86        | 42         |
| 01100110      | 102        | 102        | 102        | 102        | -26        |

**Ejercicio 3.**

*Calcular el rango y resolución de un sistema de punto fijo en BSS con 7 bits de parte entera y 3 de fraccionaria y de un sistema de punto fijo en BCS con 1 bit de signo, 5 bits de parte entera y 4 de fraccionaria.*

| Sistema   | Rango               | Resolución        |
|---|---------------------|-------------------|
| BSS - 7 bits parte entera y 3 bits parte fraccionaria                 | [0; 127,875]        | $2^{-3} = 0,125$  |
| BCS - 1 bit de signo, 5 bits parte entera y 4 bits parte fraccionaria | [-31,9375; 31,9375] | $2^{-4} = 0,0625$ |

**Ejercicio 4.**

Representar los siguientes números en los sistemas del Ejercicio 3. Si no es posible, obtener una representación exacta, indicar cuál es la más próxima y calcular, en ese caso, el error cometido. Si el número a representar está fuera del rango del sistema, señalar que ese número “NO SE PUEDE REPRESENTAR”.

| Número  | BSS - 7 bits parte entera<br>y 3 bits parte<br>fraccionaria | BCS - 1 bit de signo, 5<br>bits parte entera y 4 bits<br>parte fraccionaria |
|---------|---|---|
| 7       | 0000111 000   | 0 00111 0000  |
| 15,125  | 0001111 001   | 0 01111 0010  |
| 2,2     | 0000010 010   | 0 00010 0011  |
| 8,001   | 0001000 000   | 0 01000 0000  |
| 123,25  | 1111011 010   | ---   |
| 50,5    | 0110010 100   | ---   |
| 120     | 1111000 000   | ---   |
| 1,2     | 0000001 010   | 0 00001 0011  |
| 1,25    | 0000001 010   | 0 00001 0100  |
| 35      | 0100011 000   | ---   |
| -1,25   | ---   | 1 00001 0100  |
| 1,0625  | 0000001 001   | 0 00001 0001  |
| -1,5625 | ---   | 1 00001 1001  |
| -35,5   | ---   | ---   |

**Ejercicio 5.**

*Interpretar las siguientes cadenas en los sistemas del Ejercicio 3.*

| <b>Cadena</b> | <b>BSS - 7 bits parte entera y 3 bits parte fraccionaria</b> | <b>BCS - 1 bit de signo, 5 bits parte entera y 4 bits parte fraccionaria</b> |
|---------------|--|--|
| 0000000000    | 0  | 0  |
| 0101010101    | 42,625   | 21,3125  |
| 1000000000    | 64   | 0  |
| 1111111110    | 127,75   | -31,875  |
| 1111111111    | 127,875  | -31,9375   |
| 1010101010    | 85,25  | -10,625  |
| 0111111111    | 63,875   | 31,9375  |
| 0110110110    | 54,75  | 27,375   |

**Ejercicio 6.**

Representar los números 0, 1, 3, 8, 12, 13, 22, 35, 99, 100 y 1255 en los sistemas BCD y BCD empaquetado. Describir, con el mayor nivel de detalle posible, un procedimiento para calcular sumas en BCD. Sin considerar representación de signo, realizar las siguientes operaciones en BCD:  $32 + 45$ ;  $22 + 89$ ;  $1307 + 708$ .

| Número | BCD                                    | BCD empaquetado               |
|--------|--|-------------------------------|
| 0      | 11000000                               | 00001100                      |
| 1      | 11000001                               | 00011100                      |
| 3      | 11000011                               | 00111100                      |
| 8      | 11001000                               | 10001100                      |
| 12     | 11110001 11000010                      | 00000001 00011100             |
| 13     | 11110001 11000011                      | 00000001 00111100             |
| 22     | 11110010 11000010                      | 00000010 00101100             |
| 35     | 11110011 11000101                      | 00000011 01011100             |
| 99     | 11111001 11001001                      | 00001001 10011100             |
| 100    | 11110001 11110000<br>11000000          | 00010000 00001100             |
| 1255   | 11110001 11110010<br>11110101 11000101 | 00000001 00100101<br>01011100 |

Cuando la suma de los dos dígitos da mayor a 9, hay que generar el “acarreo” porque hay seis combinaciones no usadas. Entonces, cuando la suma de los dígitos es mayor a 9, hay que sumar 6 en ese dígito.

| Suma                | Operación   | Resultado           |
|---------------------|---|---------------------|
| $32 + 45 = 77$      | 0011 0010<br>+ 0100 0101  | 0111 0111           |
| $22 + 89 = 111$     | 0010 0010<br>+ 1000 1001 =<br>1010 1011<br>+ 0110 0110  | 0001 0001 0001      |
| $1307 + 708 = 2015$ | 0001 0011 0000 0111<br>+ 0000 0111 0000 1000<br>=<br>0001 1010 0000 1111<br>+ 0000 0110 0000 0110 | 0010 0000 0001 0101 |

**Ejercicio 7.**

*Escribir los números 13160, 2988, 927 y 87127 en los sistemas BCD, BCD empaquetado y BSS. Observar la cantidad de bits necesarios. ¿Qué conclusiones se saca respecto de las ventajas y desventajas del sistema BCD sobre BSS?*

| Número | BCD  | BCD empaquetado                  | BSS               |
|--------|--|----------------------------------|-------------------|
| 13160  | 11110001<br>11110011<br>11110001<br>11110110<br>11000000 | 00010011<br>00010110<br>00001100 | 11001101101000    |
| 2988   | 11110010<br>11111001<br>11111000<br>11001000             | 00000010<br>10011000<br>10001100 | 101110101100      |
| 927    | 11111001<br>11110010<br>11000111                         | 10010010<br>01111100             | 1110011111        |
| 87127  | 11111000<br>11110111<br>11110001<br>11110010<br>11000111 | 10000111<br>00010010<br>01111100 | 10101010001010111 |

Las conclusiones que se saca respecto de las ventajas y desventajas del sistema BCD sobre BSS son que, en aquél, es más rápida la representación, pero la cantidad de bits necesarios es mayor.

**Ejercicio 8.**

*Hacer el pasaje de binario a hexadecimal y de hexadecimal a BCH en forma directa (sin utilizar sistema decimal). ¿Por qué cree que el sistema hexadecimal es muy utilizado?*

| Binario a Hexadecimal |      |
|-----------------------|------|
| 1010000010000         | 1410 |
| 1110001011101         | 1C5D |
| 111010011001011       | 74CB |
| 1001111100100011      | 9F23 |
| 1110101011001010      | EACA |
| 101101101011010       | 5B5A |

| Hexadecimal a BCH |                  |
|-------------------|------------------|
| 2801              | 0010100000000001 |
| 1C5D              | 0001110001011101 |
| 78AB              | 0111100010101011 |
| F79A              | 1111011110011010 |
| 7EF1              | 0111111011110001 |
| 324A              | 0011001001001010 |

El sistema hexadecimal es muy utilizado porque permite representar números más grandes con menor cantidad de dígitos.

**Ejercicio 9.**

Calcular el resultado de realizar las sumas (ADD) y restas (SUB) indicadas a continuación. Calcular el valor en el que quedarán los flags luego de realizada cada operación, de acuerdo a que haya habido acarreo (flag C, de Carry) o se haya producido borrow (flag B, es el mismo que C pero en la resta), o que el resultado sea cero en todos sus bits (flag Z, de Zero), se haya producido desbordamiento (flag V, de overflow), o dé un resultado negativo (flag N, de Negative).

Recordar que:

$0+0=0$  con  $C=0$ ;  $1+0=1$  con  $C=0$ ;  $0-0=0$  con  $B=0$ ;  $1-1=0$  con  $B=0$ .

$0+1=1$  con  $C=0$ ;  $1+1=0$  con  $C=1$ ;  $1-0=1$  con  $B=0$ ;  $0-1=1$  con  $B=1$ .

También, tendremos casos de exceso en el rango de representación (llamado overflow) si a un número positivo se le suma otro positivo y da un resultado negativo ó a un número negativo se le suma otro negativo y da uno positivo ó a un número positivo se le resta otro negativo y da uno negativo ó a un número negativo se le resta otro positivo y da uno positivo. En todos estos casos de errores en la operación aritmética, se advierte el error, pues la ALU encenderá (pondrá en 1) el flag de overflow ( $V=1$ ). Es de hacer notar que el flag V se encenderá aunque se sumen números sin signo (en BSS). La interpretación de los flags corre por cuenta del programador.

| Operación              | Resultado | Z (zero) | N (negative) | C (carry) | V (overflow) |
|------------------------|-----------|----------|--------------|-----------|--------------|
| 00011101<br>+ 00011011 | 00111000  | 0        | 0            | 0         | 0            |
| 01110000<br>+ 11110001 | 01100001  | 0        | 0            | 1         | 0            |
| 10011101<br>+ 01110010 | 00001111  | 0        | 0            | 1         | 0            |
| 01001100<br>+ 01110000 | 10111100  | 0        | 1            | 0         | 1            |
| 01110110<br>+ 01110001 | 11100111  | 0        | 1            | 0         | 1            |
| 11001100<br>+ 11110000 | 10111100  | 0        | 1            | 1         | 0            |
| 10111001<br>+ 11100011 | 10011100  | 0        | 1            | 1         | 0            |
| 10000000<br>+ 10000000 | 00000000  | 1        | 0            | 1         | 1            |
| 00111010<br>+ 00001111 | 01001001  | 0        | 0            | 0         | 0            |
| 00000000<br>+ 10000000 | 10000000  | 0        | 1            | 0         | 0            |
| 00011101<br>- 00011011 | 00000010  | 0        | 0            | 0         | 0            |
| 01110000<br>- 11110001 | 01111111  | 0        | 0            | 1         | 0            |



|                        |          |   |   |   |   |
|------------------------|----------|---|---|---|---|
| 10011101<br>- 01110010 | 00101011 | 0 | 0 | 0 | 1 |
| 01001100<br>- 01110000 | 11011100 | 0 | 1 | 1 | 0 |
| 01110110<br>- 01110001 | 00000101 | 0 | 0 | 0 | 0 |
| 11001100<br>- 11110000 | 11011100 | 0 | 1 | 1 | 0 |
| 10111001<br>- 11100011 | 11010110 | 0 | 1 | 1 | 0 |
| 10000000<br>- 10000000 | 00000000 | 0 | 0 | 0 | 0 |
| 00111010<br>- 00001111 | 00101011 | 0 | 0 | 0 | 0 |
| 00000000<br>- 10000000 | 10000000 | 0 | 1 | 1 | 1 |

**Ejercicio 10.**

Suponer que los operandos del ejercicio anterior (Ejercicio 9) eran números representados en BSS, BCS, Ca1, Ca2 y Exceso2 (todos para cada sistema de representación). Verificar la correctitud del resultado interpretando el resultado obtenido y comparando con el resultado esperado. En caso de que la operación haya dado resultado incorrecto, indicar la posible cadena de bits que representa el resultado correcto.

| Operación              | Resultado | BSS         | BCS                | Ca1               | Ca2                | Ex2                 |
|------------------------|-----------|-------------|--------------------|-------------------|--------------------|---------------------|
| 00011101<br>+ 00011011 | 00111000  | 29+27=56    | 29+27=56           | 29+27=56          | 29+27=56           | -99+(-101)=-<br>72  |
| 01110000<br>+ 11110001 | 01100001  | 112+241=97  | 112+(-<br>113)=97  | 112+(-<br>14)=97  | 112+(-<br>15)=97   | -16+113=-31         |
| 10011101<br>+ 01110010 | 00001111  | 157+114=15  | -29+114=15         | -98+114=15        | -99+114=15         | 29+(-14)=-<br>113   |
| 01001100<br>+ 01110000 | 10111100  | 76+112=188  | 76+112=-60         | 76+112=-67        | 76+112=-68         | -52+(-<br>16)=60    |
| 01110110<br>+ 01110001 | 11100111  | 118+113=231 | 118+113=-<br>103   | 118+113=-<br>24   | 118+113=-<br>25    | -10+(-15)=-<br>103  |
| 11001100<br>+ 11110000 | 10111100  | 204+240=188 | -76+(-112)=-<br>60 | -51+(-15)=-<br>67 | -52+(-16)=-<br>68  | 76+112=60           |
| 10111001<br>+ 11100011 | 10011100  | 185+227=156 | -57+(-99)=-<br>28  | -70+(-28)=-<br>99 | -71+(-29)=-<br>100 | 57+99=28            |
| 10000000<br>+ 10000000 | 00000000  | 128+128=0   | 0+0=0              | -127+(-<br>127)=0 | -128+(-<br>128)=0  | 0+0=-128            |
| 00111010<br>+ 00001111 | 01001001  | 58+15=73    | 58+15=73           | 58+15=73          | 58+15=73           | -70+(-113)=-<br>55  |
| 00000000<br>+ 10000000 | 10000000  | 0+128=128   | 0+0=0              | 0+(-127)=-<br>127 | 0+(-128)=-<br>128  | -128+0=0            |
| 00011101<br>- 00011011 | 00000010  | 29-27=2     | 29-27=2            | 29-27=2           | 29-27=2            | -99-(-101)=-<br>126 |
| 01110000<br>- 11110001 | 01111111  | 112-241=127 | 112-(-<br>113)=127 | 112-(-<br>14)=127 | 112-(-<br>15)=127  | -16-113=-1          |
| 10011101<br>- 01110010 | 00101011  | 157-114=43  | -29-114=43         | -98-114=43        | -99-114=43         | 29-(-14)=-85        |
| 01001100<br>- 01110000 | 11011100  | 76-112=220  | 76-112=-92         | 76-112=-35        | 76-112=-36         | -52-(-16)=92        |
| 01110110<br>- 01110001 | 00000101  | 118-113=5   | 118-113=5          | 118-113=5         | 118-113=5          | -10-(-15)=-<br>123  |
| 11001100<br>- 11110000 | 11011100  | 204-240=220 | -76-(-112)=-<br>92 | -51-(-15)=-<br>35 | -52-(-16)=-<br>36  | 76-112=-92          |
| 10111001<br>- 11100011 | 11010110  | 185-227=214 | -57-(-99)=-<br>86  | -70-(-28)=-<br>41 | -71-(-29)=-<br>42  | 57-99=-86           |
| 10000000<br>- 10000000 | 00000000  | 128-128=0   | 0-0=0              | -127-(-<br>127)=0 | -128-(-<br>128)=0  | 0-0=-128            |
| 00111010<br>- 00001111 | 00101011  | 58-15=43    | 58-15=43           | 58-15=43          | 58-15=43           | -70-(-113)=-<br>85  |
| 00000000<br>- 10000000 | 10000000  | 0-128=128   | 0-0=0              | 0-(-127)=-<br>127 | 0-(-128)=-<br>128  | -128-0=0            |

**Ejercicio 11.**

*Referido al Ejercicio 9 sobre la operación ADD: Observando cuáles resultados fueron correctos y cuáles fueron incorrectos y relacionándolos con los flags, describir una regla para determinar la correctitud de la operación ADD en el sistema BSS con la mera observación de los flags (sin verificar la operación pasando por el sistema decimal). Observar que, en el ejemplo dado para BSS, los flags V y N quedan en 1 y no importan, pues se supone que se está operando con números sin signo (BSS). Si se hace lo mismo con todos los ejercicios, se observará que, en los casos en que  $C = 1$ , el resultado es incorrecto, independientemente de los demás flags.*

En el sistema BSS, la correctitud de la operación ADD depende de la bandera C (carry). Si se tiene  $C = 1$ , el resultado de la operación ADD en BSS va a ser incorrecto.

**Ejercicio 12.**

*Trabajar de forma similar al Ejercicio 10, pero con la operación SUB. Luego, tratar de descubrir reglas análogas para ADD y SUB para el sistema Ca2, basándose en los ejercicios cuya cadena resultado es diferente de la correcta y observando los flags. Observar qué flags se encienden en los casos que da incorrecto y cuáles no, como así también los que es indistinto que tengan valor uno o cero.*

En el sistema Ca2, la correctitud de la operación ADD y SUB depende de la bandera V (overflow). Si se tiene  $V=1$ , el resultado de estas operaciones en Ca2 va a ser incorrecto. Esta bandera será  $V=1$  cuando, en ADD, a un número positivo se le suma otro positivo y da un resultado negativo ó a un número negativo se le suma otro negativo y da uno positivo ó, en SUB, a un número positivo se le resta otro negativo y da uno negativo ó a un número negativo se le resta otro positivo y da uno positivo. Por otra parte, es indistinto que tengan valor uno o cero las banderas Z (zero), N (Negative) y C (carry).

**Ejercicio 13.**

Considerar, en el Ejercicio 9, que el punto o coma fraccionaria se encuentra entre el bit 2 y el 3. Interpretar el valor que tendrán las cadenas de bits que representan los operandos y los resultados como BSS y como Ca2. Observar los flags. ¿Qué se concluye?

| Operación              | Resultado | C (carry) | V (overflow) | BSS   | Ca2   |
|------------------------|-----------|-----------|--------------|-------|-------|
| 00011101<br>+ 00011011 | 00111000  | 0         | 0            | 14    | 14    |
| 01110000<br>+ 11110001 | 01100001  | 1         | 0            | 24,25 | 24,25 |
| 10011101<br>+ 01110010 | 00001111  | 1         | 0            | 3,75  | 3,75  |
| 01001100<br>+ 01110000 | 10111100  | 0         | 1            | 47    | -17   |
| 01110110<br>+ 01110001 | 11100111  | 0         | 1            | 57,75 | -7,25 |
| 11001100<br>+ 11110000 | 10111100  | 1         | 0            | 47    | -17   |
| 10111001<br>+ 11100011 | 10011100  | 1         | 0            | 39    | -25   |
| 10000000<br>+ 10000000 | 00000000  | 1         | 1            | 0     | 0     |
| 00111010<br>+ 00001111 | 01001001  | 0         | 0            | 18,25 | 18,25 |
| 00000000<br>+ 10000000 | 10000000  | 0         | 0            | 32    | -32   |
| 00011101<br>- 00011011 | 00000010  | 0         | 0            | 0,5   | 0,5   |
| 01110000<br>- 11110001 | 01111111  | 1         | 0            | 127   | 127   |
| 10011101<br>- 01110010 | 00101011  | 0         | 1            | 10,75 | 10,75 |
| 01001100<br>- 01110000 | 11011100  | 1         | 0            | 55    | -9    |
| 01110110<br>- 01110001 | 00000101  | 0         | 0            | 1,25  | 1,25  |
| 11001100<br>- 11110000 | 11011100  | 1         | 0            | 55    | -9    |
| 10111001<br>- 11100011 | 11010110  | 1         | 0            | 53,5  | -10,5 |
| 10000000<br>- 10000000 | 00000000  | 0         | 0            | 0     | 0     |
| 00111010<br>- 00001111 | 00101011  | 0         | 0            | 10,75 | 10,75 |
| 00000000<br>- 10000000 | 10000000  | 1         | 1            | 32    | -32   |

Por lo tanto, se concluye que, cuando  $C = 1$ , el resultado en BSS es incorrecto, mientras que, cuando  $V = 1$ , el resultado en Ca2 es incorrecto.

**Ejercicio 14.**

Escribir todas las cadenas de los sistemas BSS, BCS, Ca1, Ca2 y  $Ex2^{n-1}$  restringido a 4 bits. Considerar el punto (o coma fraccionaria) fijo en cada una de todas las posibles posiciones (son 5 posibilidades en total, considerando que el punto fijo puede estar colocado a la izquierda del MSB y a la derecha del LSB) y obtener el rango y resolución de cada uno de los sistemas de punto fijo resultantes. ¿Cuántas cadenas se pueden escribir en cada caso? ¿Cuántos números se pueden representar en los distintos sistemas?

Tabla 1 (Punto fijo en posición 1, antes del primer bit):

| Cadena     | BSS       | BCS       | Ca1       | Ca2       | $Ex2^{n-1}$ |
|------------|-----------|-----------|-----------|-----------|-------------|
| 0000       | 0         | 0         | 0         | 0         | -8          |
| 0001       | 1         | 1         | 1         | 1         | -7          |
| 0010       | 2         | 2         | 2         | 2         | -6          |
| 0011       | 3         | 3         | 3         | 3         | -5          |
| 0100       | 4         | 4         | 4         | 4         | -4          |
| 0101       | 5         | 5         | 5         | 5         | -3          |
| 0110       | 6         | 6         | 6         | 6         | -2          |
| 0111       | 7         | 7         | 7         | 7         | -1          |
| 1000       | 8         | 0         | -7        | -8        | 0           |
| 1001       | 9         | -1        | -6        | -7        | 1           |
| 1010       | 10        | -2        | -5        | -6        | 2           |
| 1011       | 11        | -3        | -4        | -5        | 3           |
| 1100       | 12        | -4        | -3        | -4        | 4           |
| 1101       | 13        | -5        | -2        | -3        | 5           |
| 1110       | 14        | -6        | -1        | -2        | 6           |
| 1111       | 15        | -7        | 0         | -1        | 7           |
| Rango      | [0; 15]   | [-7; 7]   | [-7; 7]   | [-8; 7]   | [-8; 7]     |
| Resolución | $2^0 = 1$ | $2^0 = 1$ | $2^0 = 1$ | $2^0 = 1$ | $2^0 = 1$   |

Tabla 2 (Punto fijo en posición 2, después del primer bit):

| Cadena | BSS | BCS  | Ca1  | Ca2  | $Ex2^{n-1}$ |
|--------|-----|------|------|------|-------------|
| 0000   | 0   | 0    | 0    | 0    | -4          |
| 0001   | 0,5 | 0,5  | 0,5  | 0,5  | -3,5        |
| 0010   | 1   | 1    | 1    | 1    | -3          |
| 0011   | 1,5 | 1,5  | 1,5  | 1,5  | -2,5        |
| 0100   | 2   | 2    | 2    | 2    | -2          |
| 0101   | 2,5 | 2,5  | 2,5  | 2,5  | -1,5        |
| 0110   | 3   | 3    | 3    | 3    | -1          |
| 0111   | 3,5 | 3,5  | 3,5  | 3,5  | -0,5        |
| 1000   | 4   | 0    | -3,5 | -4   | 0           |
| 1001   | 4,5 | -0,5 | -3   | -3,5 | 0,5         |
| 1010   | 5   | -1   | -2,5 | -3   | 1           |
| 1011   | 5,5 | -1,5 | -2   | -2,5 | 1,5         |
| 1100   | 6   | -2   | -1,5 | -2   | 2           |

|            |              |              |              |              |              |
|------------|--------------|--------------|--------------|--------------|--------------|
| 1101       | 6,5          | -2,5         | -1           | -1,5         | 2,5          |
| 1110       | 7            | -3           | -0,5         | -1           | 3            |
| 1111       | 7,5          | -3,5         | 0            | -0,5         | 3,5          |
| Rango      | [0; 7,5]     | [-3,5; 3,5]  | [-3,5; 3,5]  | [-4; 3,5]    | [-4; 3,5]    |
| Resolución | $2^{-1}=0,5$ | $2^{-1}=0,5$ | $2^{-1}=0,5$ | $2^{-1}=0,5$ | $2^{-1}=0,5$ |

Tabla 3 (Punto fijo en posición 3, después del segundo bit):

| Cadena     | BSS           | BCS           | Ca1           | Ca2           | $Ex2^{n-1}$   |
|------------|---------------|---------------|---------------|---------------|---------------|
| 0000       | 0             | 0             | 0             | 0             | -2            |
| 0001       | 0,25          | 0,25          | 0,25          | 0,25          | -1,75         |
| 0010       | 0,5           | 0,5           | 0,5           | 0,5           | -1,5          |
| 0011       | 0,75          | 0,75          | 0,75          | 0,75          | -1,25         |
| 0100       | 1             | 1             | 1             | 1             | -1            |
| 0101       | 1,25          | 1,25          | 1,25          | 1,25          | -0,75         |
| 0110       | 1,5           | 1,5           | 1,5           | 1,5           | -0,5          |
| 0111       | 1,75          | 1,75          | 1,75          | 1,75          | -0,25         |
| 1000       | 2             | 0             | -1,75         | -2            | 0             |
| 1001       | 2,25          | -0,25         | -1,5          | -1,75         | 0,25          |
| 1010       | 2,5           | -0,5          | -1,25         | -1,5          | 0,5           |
| 1011       | 2,75          | -0,75         | -1            | -1,25         | 0,75          |
| 1100       | 3             | -1            | -0,75         | -1            | 1             |
| 1101       | 3,25          | -1,25         | -0,5          | -0,75         | 1,25          |
| 1110       | 3,5           | -1,5          | -0,25         | -0,5          | 1,5           |
| 1111       | 3,75          | -1,75         | 0             | -0,25         | 1,75          |
| Rango      | [0; 3,75]     | [-1,75; 1,75] | [-1,75; 1,75] | [-2; 1,75]    | [-2; 1,75]    |
| Resolución | $2^{-2}=0,25$ | $2^{-2}=0,25$ | $2^{-2}=0,25$ | $2^{-2}=0,25$ | $2^{-2}=0,25$ |

Tabla 4 (Coma en posición 4, después del tercer bit):

| Cadena | BSS   | BCS    | Ca1    | Ca2    | $Ex2^{n-1}$ |
|--------|-------|--------|--------|--------|-------------|
| 0000   | 0     | 0      | 0      | 0      | -1          |
| 0001   | 0,125 | 0,125  | 0,125  | 0,125  | -0,875      |
| 0010   | 0,25  | 0,25   | 0,25   | 0,25   | -0,75       |
| 0011   | 0,375 | 0,375  | 0,375  | 0,375  | -0,625      |
| 0100   | 0,5   | 0,5    | 0,5    | 0,5    | -0,5        |
| 0101   | 0,625 | 0,625  | 0,625  | 0,625  | -0,375      |
| 0110   | 0,75  | 0,75   | 0,75   | 0,75   | -0,25       |
| 0111   | 0,875 | 0,875  | 0,875  | 0,875  | -0,125      |
| 1000   | 1     | 0      | -0,875 | -1     | 0           |
| 1001   | 1,125 | -0,125 | -0,75  | -0,875 | 0,125       |
| 1010   | 1,25  | -0,25  | -0,625 | -0,75  | 0,25        |
| 1011   | 1,375 | -0,375 | -0,5   | -0,625 | 0,375       |
| 1100   | 1,5   | -0,5   | -0,375 | -0,5   | 0,5         |
| 1101   | 1,625 | -0,625 | -0,25  | -0,375 | 0,625       |
| 1110   | 1,75  | -0,75  | -0,125 | -0,25  | 0,75        |
| 1111   | 1,875 | -0,875 | 0      | -0,125 | 0,875       |



|            |                  |                    |                    |                  |                  |
|------------|------------------|--------------------|--------------------|------------------|------------------|
| Rango      | [0; 1,875]       | [-0,875;<br>0,875] | [-0,875;<br>0,875] | [-1; 0,875]      | [-1; 0,875]      |
| Resolución | $2^{-3} = 0,125$ | $2^{-3} = 0,125$   | $2^{-3} = 0,125$   | $2^{-3} = 0,125$ | $2^{-3} = 0,125$ |

Tabla 5 (Coma en posición 5, después del cuarto bit):

| Cadena     | BSS               | BCS                | Ca1                  | Ca2               | $Ex2^{n-1}$       |
|------------|-------------------|--------------------|----------------------|-------------------|-------------------|
| 0000       | 0                 | 0                  | 0                    | 0                 | -0,5              |
| 0001       | 0,0625            | 0,125              | 0,0625               | 0,0625            | -0,4375           |
| 0010       | 0,125             | 0,25               | 0,125                | 0,125             | -0,375            |
| 0011       | 0,1875            | 0,375              | 0,1875               | 0,1875            | -0,3125           |
| 0100       | 0,25              | 0,5                | 0,25                 | 0,25              | -0,25             |
| 0101       | 0,3125            | 0,625              | 0,3125               | 0,3125            | -0,1875           |
| 0110       | 0,375             | 0,75               | 0,375                | 0,375             | -0,125            |
| 0111       | 0,4375            | 0,875              | 0,4375               | 0,4375            | -0,0625           |
| 1000       | 0,5               | 0                  | -0,4375              | -0,5              | 0                 |
| 1001       | 0,5625            | -0,125             | -0,375               | -0,4375           | 0,0625            |
| 1010       | 0,625             | -0,25              | -0,3125              | -0,375            | 0,125             |
| 1011       | 0,6875            | -0,375             | -0,25                | -0,3125           | 0,1875            |
| 1100       | 0,75              | -0,5               | -0,1875              | -0,25             | 0,25              |
| 1101       | 0,8125            | -0,625             | -0,125               | -0,1875           | 0,3125            |
| 1110       | 0,875             | -0,75              | -0,0625              | -0,125            | 0,375             |
| 1111       | 0,9375            | -0,875             | 0                    | -0,0625           | 0,4375            |
| Rango      | [0; 0,9375]       | [-0,875;<br>0,875] | [-0,4375;<br>0,4375] | [-0,5;<br>0,4375] | [-0,5;<br>0,4375] |
| Resolución | $2^{-4} = 0,0625$ | $2^{-3} = 0,125$   | $2^{-4} = 0,0625$    | $2^{-4} = 0,0625$ | $2^{-4} = 0,0625$ |

Por lo tanto, las cadenas que se pueden escribir, en cada caso (BSS, BCS, Ca1, Ca2,  $Ex2^{n-1}$ ), dadas las 5 posibles posiciones del punto fijo, son 16. Por otra parte, la cantidad de números que se pueden representar en estos distintos sistemas son 48, 47, 47, 48 y 48, respectivamente.

**Ejercicio 15.**

*Definir el sistema Exceso a  $M$  (donde  $M$  es un entero cualquiera).*

El sistema Exceso a  $M$  es un sistema de representación numérica que utiliza un desplazamiento en la escala de números positivos para permitir la representación de números negativos. En este sistema, se asigna un valor base  $M$  y se utiliza un código que representa números positivos con valores entre 0 y  $M-1$ .

Para representar números negativos en el sistema Exceso a  $M$ , se utiliza un código que se desplaza en una cantidad fija  $M$ , de modo que el valor negativo se convierte en un valor positivo. Por lo tanto, el valor representado en el sistema Exceso a  $M$  es igual al valor real del número más  $M$ .

En general, en el sistema Exceso a  $M$ , se representa el número  $x$  como  $x + M$  en código binario, lo que permite la representación de números negativos mediante un simple desplazamiento en la escala de números positivos.

**Ejercicio 16.**

*Describir mecanismos para sumar y restar en BCS, Ca1 y Exceso, en base al análisis de los resultados y flags del Ejercicio 9, realizando la interpretación de los operandos y resultados en los distintos sistemas de representación citados. Observar de qué manera (qué operaciones deberían realizarse y en qué caso) se llegaría al resultado correcto.*

El resultado de sumar y restar es incorrecto:

- En BCS, cuando hay  $C=1$  o  $V=1$ , pero no ambas. Se llegaría al resultado correcto agregando un bit.
- En Ca1, cuando hay  $C=1$  o  $V=1$  o ambas. Se llegaría al resultado correcto agregando un bit.
- En Ex2, siempre. Cuando  $C=0$ , se llegaría al resultado correcto contemplando excesos distintos para los operandos y para el resultado, es decir, contemplando en los operandos, un exceso de  $2^{n+1}$  y, en el resultado, un exceso de  $2^n$  en la suma y ningún exceso en la resta.

**Ejercicio 17.**

*Interpretar las siguientes cadenas descriptas en sistema Ca2. ¿Qué pasa en el caso (e)?*

| <b>Cadena</b> | <b>Interpretación</b> |
|---------------|-----------------------|
| 00100110      | 38                    |
| 11011000      | -40                   |
| 00111000      | 56                    |
| 00000000      | 0                     |
| 10000000      | -128                  |

**Ejercicio 18.**

*Interpretar las siguientes cadenas descriptas en sistema  $Ex2^{n-1}$  con  $n=8$ . ¿Qué pasa en el caso (e)?*

| Cadena   | Interpretación |
|----------|----------------|
| 10100110 | 38             |
| 01011000 | -40            |
| 10111000 | 56             |
| 10000000 | 0              |
| 00000000 | -128           |

## Sistema de Numeración en Punto Flotante

**Objetivos de la práctica:** que el alumno domine los tópicos de sistemas de numeración referidos a las representaciones en punto flotante, tales como:

- Representación e interpretación.
- Operaciones aritméticas.
- IEEE 754.

**Bibliografía:**

- "Organización y Arquitectura de Computadores" de W. Stalling, capítulo 8.
- Apunte 2 de la Cátedra, "Sistemas de numeración: Punto flotante".

1. Considerando el sistema de Punto Flotante cuya mantisa es fraccionaria, con 6 bits, está expresada en BSS (en el inciso a) o BCS (en el inciso b) y su exponente en BCS con 4 bits, escriba el significado de las siguientes cadenas de bits (mantisa a la izquierda):

| Cadena     | a) Mantisa en BSS | b) Mantisa en BCS |
|------------|-------------------|-------------------|
| 0101110110 |                   |                   |
| 0000010000 |                   |                   |
| 0000111001 |                   |                   |
| 1111111111 |                   |                   |
| 0000000000 |                   |                   |
| 0000001111 |                   |                   |
| 1111110000 |                   |                   |
| 1000000000 |                   |                   |
| 0000011111 |                   |                   |

2. Dado un sistema de Punto Flotante cuya mantisa es fraccionaria, está expresada en BCS con 5 bits y su exponente en BSS con 3 bits, interprete las siguientes cadenas del considerando que la mantisa esta sin normalizar, normalizada, o normalizada con bit implícito. Identifique aquellas cadenas que no pueden ser interpretadas y mencione porqué.

| Cadena   | Sin normalizar | Normalizada | Normalizada con Bit Implícito |
|----------|----------------|-------------|-------------------------------|
| 01000111 |                |             |                               |
| 11000011 |                |             |                               |
| 00000000 |                |             |                               |
| 11111111 |                |             |                               |

3. Calcule rango y resolución en extremos inferior negativo, superior negativo, inferior positivo y superior positivo para los siguientes sistemas de representación en punto flotante:

- Mantisa fraccionaria en BSS de 8 bits y exponente en BSS 4 bits
- Mantisa fraccionaria normalizada en BSS de 15 bits y exponente en CA1 10 bits
- Mantisa fraccionaria normalizada con bit implícito en BCS de 15 bits y exponente en Exceso 5 bits
- Mantisa fraccionaria normalizada con bit implícito en BCS de N bits y exponente en CA2 de M bits

Observe que:

- En las mantisas BSS no se puede expresar números negativos, con lo que aun con exponente negativo expresaremos un número positivo por un factor de escala menor a 1, pero también positivo. Ejemplo:  $2 \times 2^{-4} = 0,125$ .
- Las mantisas fraccionarias suponen el punto al principio de la mantisa.
- Los exponentes negativos indican factores de escala menores a 1 que mejoran la resolución.
- Mantisa normalizada implica que empieza con 1, o sea mantisa mínima 0,1 para la fraccionaria, igual a 0,5 en decimal. Esto hace que no se pueda representar el 0.
- Mantisa normalizada con bit implícito, significa agregar un 1 al principio de la misma al interpretarla. Ejemplo: 00000 se interpreta 0,100000, o 0,5 en base 10.

4. Dado un sistema de Punto Flotante cuya mantisa es fraccionaria, está expresada en BCS con 10 bits y su exponente en CA2 con 5 bits, obtenga la representación de los siguientes números, considerando que la mantisa esta sin normalizar, normalizada, o normalizada con bit implícito

| Cadena    | Sin normalizar | Normalizada | Normalizada con Bit Implícito |
|-----------|----------------|-------------|-------------------------------|
| 0         |                |             |                               |
| 1         |                |             |                               |
| 9         |                |             |                               |
| -5,0625   |                |             |                               |
| 34000,5   |                |             |                               |
| 0,015625  |                |             |                               |
| Nº máximo |                |             |                               |
| Nº mínimo |                |             |                               |

# Organización de Computadoras 2023

- [illegible]

## **Trabajo Práctico N° 2:** **Sistema de Numeración en Punto Flotante.**

### **Ejercicio 1.**

Considerando el sistema de Punto Flotante cuya mantisa es fraccionaria, con 6 bits, está expresada en BSS (en el inciso a) o BCS (en el inciso b) y su exponente en BCS con 4 bits, escribir el significado de las siguientes cadenas de bits (mantisa a la izquierda):

| Cadena      | (a) Mantisa en BSS   | (b) Mantisa en BCS   |
|-------------|--|--|
| 010111 0110 | $010111 * 2^{0110} = (2^{-2} + 2^{-4} + 2^{-5} + 2^{-6}) * 2^6 = 2^4 + 2^2 + 2 + 1 = 23$                   | $010111 * 2^{0110} = (2^{-1} + 2^{-3} + 2^{-4} + 2^{-5}) * 2^6 = 2^5 + 2^3 + 2^2 + 2 = 46$                 |
| 000001 0000 | $000001 * 2^{0000} = 2^{-6} * 2^0 = \frac{1}{64} * 1 = \frac{1}{64}$                                       | $000001 * 2^{0000} = 2^{-5} * 2^0 = \frac{1}{32} * 1 = \frac{1}{32}$                                       |
| 000011 1001 | $000011 * 2^{1001} = (2^{-5} + 2^{-6}) * 2^{-1} = 2^{-6} + 2^{-7} = \frac{1}{128} (2 + 1) = \frac{3}{128}$ | $000011 * 2^{1001} = (2^{-4} + 2^{-5}) * 2^{-1} = 2^{-5} + 2^{-6} = \frac{1}{64} (2 + 1) = \frac{3}{64}$   |
| 111111 1111 | $111111 * 2^{1111} = (1 - 2^{-6}) * 2^{-7} = 2^{-7} - 2^{-13} = \frac{1}{8192} (64 + 1) = \frac{65}{8192}$ | $111111 * 2^{1111} = (1 - 2^{-5}) * 2^{-7} = 2^{-7} - 2^{-12} = \frac{1}{4096} (32 + 1) = \frac{33}{4096}$ |
| 000000 0000 | $000000 * 2^{0000} = 0 * 2^0 = 0 * 1 = 0$  | $000000 * 2^{0000} = 0 * 2^0 = 0 * 1 = 0$  |
| 000000 1111 | $000000 * 2^{1111} = 0 * 2^{-7} = 0$   | $000000 * 2^{1111} = 0 * 2^{-7} = 0$   |
| 111111 0000 | $111111 * 2^{0000} = (1 - 2^{-6}) * 2^0 = (1 - \frac{1}{64}) * 1 = \frac{63}{64}$                          | $111111 * 2^{0000} = -(1 - 2^{-5}) * 2^0 = -(1 - \frac{1}{32}) * 1 = -\frac{31}{32}$                       |
| 100000 0000 | $100000 * 2^{0000} = 2^{-1} * 2^0 = \frac{1}{2} * 1 = \frac{1}{2}$   | $100000 * 2^{0000} = -2^{-1} * 2^0 = -\frac{1}{2} * 1 = -\frac{1}{2}$                                      |
| 000001 1111 | $000001 * 2^{1111} = 2^{-6} * 2^{-7} = 2^{-13} = \frac{1}{8192}$   | $000001 * 2^{1111} = 2^{-5} * 2^{-7} = 2^{-12} = \frac{1}{4096}$   |



**Ejercicio 2.**

Dado un sistema de Punto Flotante cuya mantisa es fraccionaria, está expresada en BCS con 5 bits y su exponente en BSS con 3 bits, interpretar las siguientes cadenas considerando que la mantisa está sin normalizar, normalizada o normalizada con bit implícito. Identificar aquellas cadenas que no pueden ser interpretadas y mencionar por qué.

| Cadena    | Sin normalizar   | Normalizada  | Normalizada con bit implícito   |
|-----------|--|--|---|
| 01000 111 | $0\ 1000 * 2^{111} = 2^{-1} * 2^7 = 2^6 = 64$                                | $0\ 1000 * 2^{111} = 2^{-1} * 2^7 = 2^6 = 64$                                | $0\ [1]1000 * 2^{111} = (2^{-1} + 2^{-2}) * 2^7 = 2^6 + 2^5 = 64 + 32 = 96$     |
| 11000 011 | $1\ 1000 * 2^{011} = -2^{-1} * 2^3 = -2^2 = -4$                              | $1\ 1000 * 2^{011} = -2^{-1} * 2^3 = -2^2 = -4$                              | $1\ [1]1000 * 2^{011} = -(2^{-1} + 2^{-2}) * 2^3 = -(2^2 + 2) = -(4 + 2) = -6$  |
| 00000 000 | $0\ 0000 * 2^{000} = 0 * 2^0 = 0 * 1 = 0$                                    | ---  | $0\ [1]0000 * 2^{000} = 2^{-1} * 2^0 = 0,5 * 1 = 0,5$                           |
| 11111 111 | $1\ 1111 * 2^{111} = -(1 - 2^{-4}) * 2^7 = -(2^7 - 2^3) = -(128 - 8) = -120$ | $1\ 1111 * 2^{111} = -(1 - 2^{-4}) * 2^7 = -(2^7 - 2^3) = -(128 - 8) = -120$ | $1\ [1]1111 * 2^{111} = -(1 - 2^{-5}) * 2^7 = -(2^7 - 2^2) = -(128 - 4) = -124$ |

**Ejercicio 3.**

Calcular rango y resolución en extremos inferior negativo, superior negativo, inferior positivo y superior positivo para los siguientes sistemas de representación en punto flotante:

Observar que:

- En las mantisas BSS, no se puede expresar números negativos, con lo que, aún con exponente negativo, expresaremos un número positivo por un factor de escala menor a 1, pero también positivo. Ejemplo:  $2 * 2^{-4} = 0,125$ .
- Las mantisas fraccionarias suponen el punto al principio de la mantisa.
- Los exponentes negativos indican factores de escala menores a 1 que mejoran la resolución.
- Mantisa normalizada implica que empieza con 1, o sea mantisa mínima 0,1 para la fraccionaria, igual a 0,5 en decimal. Esto hace que no se pueda representar el 0.
- Mantisa normalizada con bit implícito, significa agregar un 1 al principio de la misma al interpretarla. Ejemplo: 00000 se interpreta 0,100000 o 0,5 en base 10.

(a) Mantisa fraccionaria en BSS de 8 bits y exponente en BSS 4 bits.

$$\text{Rango} = [00000000 * 2^{0000}, 11111111 * 2^{1111}]$$

$$\text{Rango} = [0 * 2^0; (1 - 2^{-8}) * 2^{15}]$$

$$\text{Rango} = [0 * 1; (2^{15} - 2^7)]$$

$$\text{Rango} = [0; 32640].$$

$$\text{Resolución en el extremo inferior} = 2^{-8} * 2^0 = 2^{-8} * 1 = 2^{-8} = \frac{1}{256}.$$

$$\text{Resolución en el extremo superior} = 2^{-8} * 2^{15} = 2^7 = 128.$$

(b) Mantisa fraccionaria normalizada en BSS de 15 bits y exponente en CAI 10 bits.

$$\text{Rango} = [100000000000000 * 2^{1000000000}, 111111111111111 * 2^{0111111111}]$$

$$\text{Rango} = [0,5 * 2^{-511}; (1 - 2^{-15}) * 2^{511}]$$

$$\text{Rango} = [0,5 * 2^{-511}; (2^{511} - 2^{496})].$$

$$\text{Resolución en el extremo inferior} = 2^{-15} * 2^{-511} = 2^{-526}.$$

$$\text{Resolución en el extremo superior} = 2^{-15} * 2^{511} = 2^{496}.$$

(c) Mantisa fraccionaria normalizada con bit implícito en BCS de 15 bits y exponente en Exceso 5 bits.

$$\text{Rango negativo} = [1 [1]111111111111111 * 2^{11111}, 1 [1]000000000000000 * 2^{00000}]$$

$$\text{Rango negativo} = [-(1 - 2^{-15}) * 2^{15}; -2^{-1} * 2^{-16}]$$

$$\text{Rango negativo} = [-(2^{15} - 1); -2^{-17}].$$

Rango positivo=  $[0 [1]0000000000000000 * 2^{000000}; 0 [1]1111111111111111 * 2^{111111}]$

Rango positivo=  $[2^{-1} * 2^{-16}; (1 - 2^{-15}) * 2^{15}]$

Rango positivo=  $[2^{-17}; (2^{15} - 1)]$ .

Resolución en el extremo superior negativo / extremo inferior positivo=  $2^{-15} * 2^{-16} = 2^{-31}$ .

Resolución en el extremo inferior negativo / extremo superior positivo=  $2^{-15} * 2^{15} = 1$ .

**(d)** *Mantisa fraccionaria normalizada con bit implícito en BCS de N bits y exponente en CA2 de M bits.*

Rango negativo=  $[-(1 - 2^{-N}) * 2^{2^{M-1}-1}; -2^{-1} * 2^{-2^{M-1}}]$ .

Rango positivo=  $[2^{-1} * 2^{-2^{M-1}}; (1 - 2^{-N}) * 2^{2^{M-1}-1}]$ .

Resolución en el extremo superior negativo / extremo inferior positivo=  $2^{-N} * 2^{-2^{M-1}}$ .

Resolución en el extremo inferior negativo / extremo superior positivo=  $2^{-N} * 2^{2^{M-1}-1}$ .

**Ejercicio 4.**

Dado un sistema de Punto Flotante cuya mantisa es fraccionaria, está expresada en BCS con 10 bits y su exponente en CA2 con 5 bits, obtener la representación de los siguientes números, considerando que la mantisa está sin normalizar, normalizada o normalizada con bit implícito.

| Cadena        | Sin normalizar  | Normalizada   | Normalizada con bit implícito   |
|---------------|---|---|---|
| 0             | $0\ 000000000\ * 2^{00000}$   | ---   | ---   |
| 1             | $0\ 100000000\ * 2^{00001} =$<br>$0\ 010000000\ * 2^{00010} =$<br>$0\ 001000000\ * 2^{00011} \dots$ | $0\ 100000000\ * 2^{00001}$   | $0\ [1]000000000\ * 2^{00001}$  |
| 9             | $0\ 100100000\ * 2^{00100} =$<br>$0\ 010010000\ * 2^{00101} =$<br>$0\ 001001000\ * 2^{00110} \dots$ | $0\ 100100000\ * 2^{00100}$   | $0\ [1]001000000\ * 2^{00100}$  |
| -5,0625       | $1\ 101000100\ * 2^{00011} =$<br>$1\ 010100010\ * 2^{00100} =$<br>$1\ 001010001\ * 2^{00101} \dots$ | $1\ 101000100\ * 2^{00011}$   | $1\ [1]010001000\ * 2^{00011}$  |
| 34000,5       | ---   | ---   | ---   |
| 0,015625      | $0\ 000001000\ * 2^{00000} =$<br>$0\ 000010000\ * 2^{11111} =$<br>$0\ 000100000\ * 2^{11110} \dots$ | $0\ 100000000\ * 2^{11011}$   | $0\ [1]000000000\ * 2^{11011}$  |
| Número máximo | $0\ 111111111\ * 2^{01111} = (1 - 2^{-9}) * 2^{15} = 2^{15} - 2^6 = 32704$                          | $0\ 111111111\ * 2^{01111} = (1 - 2^{-9}) * 2^{15} = 2^{15} - 2^6 = 32704$      | $0\ [1]111111111\ * 2^{01111} = (1 - 2^{-10}) * 2^{15} = 2^{15} - 2^5 = 32736$      |
| Número mínimo | $1\ 111111111\ * 2^{01111} = -(1 - 2^{-9}) * 2^{15} = -(2^{15} - 2^6) = -32704$                     | $1\ 111111111\ * 2^{01111} = -(1 - 2^{-9}) * 2^{15} = -(2^{15} - 2^6) = -32704$ | $1\ [1]111111111\ * 2^{01111} = -(1 - 2^{-10}) * 2^{15} = -(2^{15} - 2^5) = -32736$ |

**Ejercicio 5.**

*Decir cómo influyen las siguientes variantes en el rango y resolución:*

**(a) Mantisa con signo y sin signo.**

Mantisa con signo (supongo mantisa entera y exponente en BCS):

$$\text{Rango} = [-(2^{M-1} - 1) * 2^{2^{E-1}-1}; (2^{M-1} - 1) * 2^{2^{E-1}-1}].$$

$$\text{Resolución en el extremo inferior} = 2^0 * 2^{-(2^{E-1}-1)}.$$

$$\text{Resolución en el extremo superior} = 2^0 * 2^{2^{E-1}-1}.$$

Mantisa sin signo (supongo mantisa entera y exponente en BCS):

$$\text{Rango} = [0; (2^M - 1) * 2^{2^{E-1}-1}].$$

$$\text{Resolución en el extremo inferior} = 2^0 * 2^{-(2^{E-1}-1)}.$$

$$\text{Resolución en el extremo superior} = 2^0 * 2^{2^{E-1}-1}.$$

**(b) Exponente con signo y sin signo.**

Exponente con signo (supongo mantisa entera y en BCS):

$$\text{Rango} = [-(2^{M-1} - 1) * 2^{2^{E-1}-1}; (2^{M-1} - 1) * 2^{2^{E-1}-1}].$$

$$\text{Resolución en el extremo inferior} = 2^0 * 2^{-(2^{E-1}-1)}.$$

$$\text{Resolución en el extremo superior} = 2^0 * 2^{2^{E-1}-1}.$$

Exponente sin signo (supongo mantisa entera y en BCS):

$$\text{Rango} = [-(2^{M-1} - 1) * 2^{2^E-1}; (2^{M-1} - 1) * 2^{2^E-1}].$$

$$\text{Resolución en el extremo inferior} = 2^0 * 2^0.$$

$$\text{Resolución en el extremo superior} = 2^0 * 2^{2^E-1}.$$

**(c) Tamaño de mantisa.**

Supongo mantisa entera en BCS y exponente en BCS:

$$\text{Rango} = [-(2^{M-1} - 1) * 2^{2^{E-1}-1}; (2^{M-1} - 1) * 2^{2^{E-1}-1}].$$

$$\text{Resolución en el extremo inferior} = 2^0 * 2^{-(2^{E-1}-1)}.$$

$$\text{Resolución en el extremo superior} = 2^0 * 2^{2^{E-1}-1}.$$

**(d) Tamaño de exponente.**

Supongo mantisa entera en BCS y exponente en BCS:

$$\text{Rango} = [-(2^{M-1} - 1) * 2^{2^{E-1}-1}; (2^{M-1} - 1) * 2^{2^{E-1}-1}].$$

$$\text{Resolución en el extremo inferior} = 2^0 * 2^{-(2^{E-1}-1)}.$$

$$\text{Resolución en el extremo superior} = 2^0 * 2^{2^{E-1}-1}.$$

*(e) Mantisa fraccionaria, fraccionaria normalizada y fraccionaria normalizada con bit implícito.*

Mantisa fraccionaria (supongo mantisa y exponente en BCS):

$$\text{Rango} = [-(1 - 2^{-(M-1)}) * 2^{2^{E-1}-1}; (1 - 2^{-(M-1)}) * 2^{2^{E-1}-1}].$$

$$\text{Resolución en el extremo inferior} = 2^{-(M-1)} * 2^{-(2^{E-1}-1)}.$$

$$\text{Resolución en el extremo superior} = 2^{-(M-1)} * 2^{2^{E-1}-1}.$$

Mantisa fraccionaria normalizada (supongo mantisa y exponente en BCS):

$$\text{Rango negativo} = [-(1 - 2^{-(M-1)}) * 2^{2^{E-1}-1}; -2^{-(M-1)} * 2^{-(2^{E-1}-1)}].$$

$$\text{Rango positivo} = [2^{-(M-1)} * 2^{-(2^{E-1}-1)}; (1 - 2^{-(M-1)}) * 2^{2^{E-1}-1}].$$

$$\text{Resolución en el extremo superior negativo / extremo inferior positivo} = 2^{-(M-1)} * 2^{-(2^{E-1}-1)}.$$

$$\text{Resolución en el extremo inferior negativo / extremo superior positivo} = 2^{-(M-1)} * 2^{2^{E-1}-1}.$$

Mantisa fraccionaria normalizada con bit implícito (supongo mantisa y exponente en BCS):

$$\text{Rango negativo} = [-(1 - 2^{-M}) * 2^{2^{E-1}-1}; -2^{-M} * 2^{-(2^{E-1}-1)}].$$

$$\text{Rango positivo} = [2^{-M} * 2^{-(2^{E-1}-1)}; (1 - 2^{-M}) * 2^{2^{E-1}-1}].$$

$$\text{Resolución en el extremo superior negativo / extremo inferior positivo} = 2^{-M} * 2^{-(2^{E-1}-1)}.$$

$$\text{Resolución en el extremo inferior negativo / extremo superior positivo} = 2^{-M} * 2^{2^{E-1}-1}.$$

**Ejercicio 6.**

Efectuar las siguientes sumas para un sistema de punto flotante con mantisa en BSS de 8 bits y exponente en BCS de 8 bits. Observar que los factores de escala deben ser los mismos, sino se sumarían dos mantisas con pesos distintos (recordar que se puede correr los unos y sumar o restar este corrimiento al exponente para obtener una cadena equivalente).

(a) 00001111 00000011 + 00001000 00000010.

Opción 1:

00001111 00000011 +  
00001000 00000010 =

00001111 00000011 +  
00000100 00000011 =

00010011 00000011 =  $19 * 2^3 = 19 * 8 = 152$ .

Opción 2:

00001111 00000011 +  
00001000 00000010 =

00011110 00000010 +  
00001000 00000010 =

00100110 00000010 =  $38 * 2^2 = 38 * 4 = 152$ .

(b) 01111111 00000000 + 11111100 10000001.

Opción 1:

01111111 00000000 +  
11111100 10000001 =

01111111 00000000 +  
01111110 00000000 =

11111101 00000000 =  $253 * 2^0 = 253 * 1 = 253$ .

Opción 2:

01111111 00000000 +  
11111100 10000001 =

11111110 10000001 +  
11111100 10000001 =

[1]11111010 10000001 =

11111101 00000000 =  $253 * 2^0 = 253 * 1 = 253$ .

(c) 00000001 00000111 + 00011100 00000000.

Opción 1:

00000001 00000111 +  
00011100 00000000 =

10000000 00000000 +  
00011100 00000000 =

10011100 00000000 =  $156 * 2^0 = 156 * 1 = 156$ .

Opción 2:

00000001 00000111 +  
00011100 00000000 =

00100000 00000010 +  
00000111 00000010 =

00100111 00000010 =  $39 * 2^2 = 39 * 4 = 156$ .



**Ejercicio 7.**

*Suponiendo que los números que no son representables se aproximan al más próximo, obtener las representaciones o aproximaciones de los números 8.625, 0.4 y 2.5 en los sistemas.*

| Número | (a) Mantisa fraccionaria normalizada de 5 bits BSS y exponente de 4 bits CA2 | (b) Mantisa fraccionaria normalizada de 10 bits BCS y exponente de 3 bits CA2 |
|--------|--|---|
| 8,625  | $10001 * 2^{0100} = 8,5$   | $0\ 111111111 * 2^{011} = 7,984375$   |
| 0,4    | $11010 * 2^{1111} = 0,40625$   | $0\ 110011010 * 2^{111} = 0,400390625$  |
| 2,5    | $10100 * 2^{0010} = 2,5$   | $0\ 101000000 * 2^{010} = 2,5$  |

**Ejercicio 8.**

Se define Error Absoluto y Error Relativo de un número  $x$  en un sistema de la siguiente forma:  $EA(x) = |x' - x|$  y  $ER(x) = \frac{EA(x)}{x}$ , donde  $x'$  es el número representable del sistema más próximo a  $x$ . Calcular los errores absolutos y relativos para los casos del ejercicio anterior.

| Número | (a)                         |                                  | (b)                                 |  |
|--------|-----------------------------|----------------------------------|-------------------------------------|--|
|        | EA                          | ER                               | EA                                  | ER                                       |
| 8,625  | $ 8,5 - 8,625  = 0,125$     | $\frac{0,125}{8,625} = 0,0145$   | $ 7,984375 - 8,625  = 0,640625$     | $\frac{0,640625}{8,625} = 0,0743$        |
| 0,4    | $ 0,40625 - 0,4  = 0,00625$ | $\frac{0,00625}{0,4} = 0,015625$ | $ 0,400390625 - 0,4  = 0,000390625$ | $\frac{0,000390625}{0,4} = 0,0009765625$ |
| 2,5    | $ 2,5 - 2,5  = 0$           | $\frac{0}{2,5} = 0$              | $ 2,5 - 2,5  = 0$                   | $\frac{0}{2,5} = 0$                      |

**Ejercicio 9.**

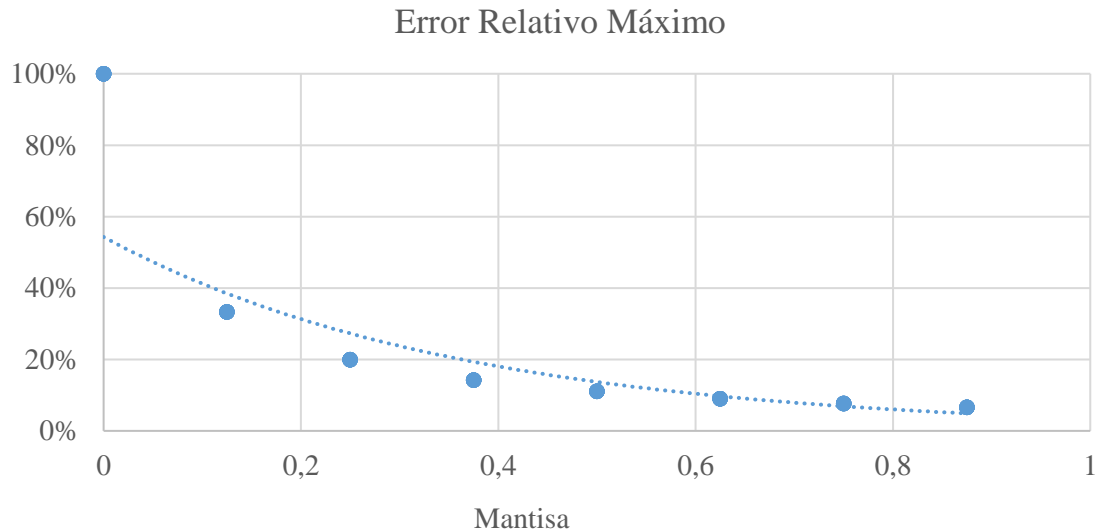
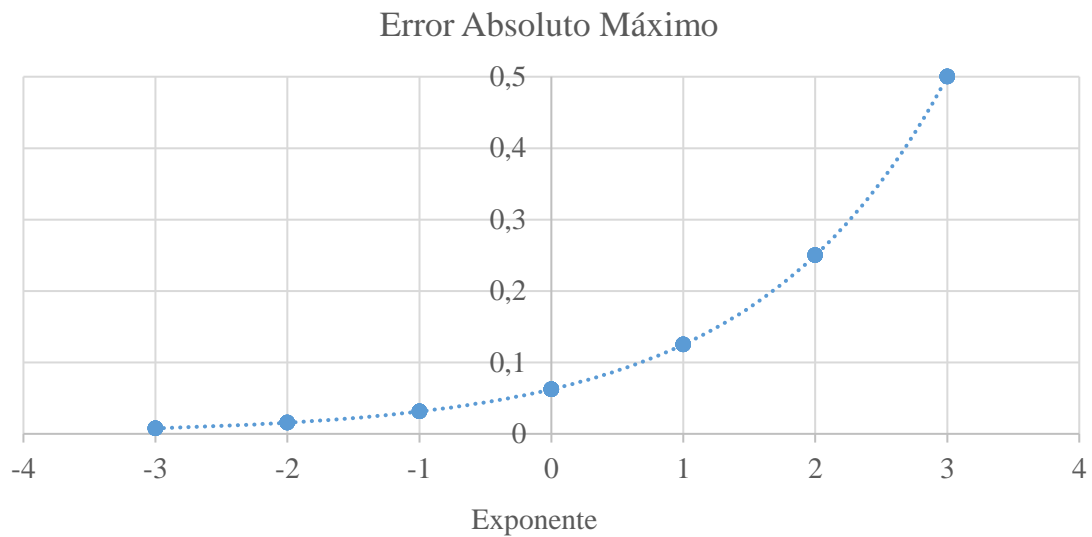
Considerando que, en los procesos de truncamiento o redondeo, la elección se basa en la representación más cercana, estimar el Error Absoluto Máximo cometido en las representaciones del ejercicio 7. Recordar que la distancia entre 2 representaciones sucesivas se conoce como resolución ( $R$ ), por lo que  $EAmáx \leq \frac{R}{2}$ .

| Número | EAmáx en (a)                                    | EAmáx en (b)   |
|--------|---|--|
| 8,625  | $2^{-1} * 2^{-5} * 2^4 = 2^{-2} = 0,25$         | ---  |
| 0,4    | $2^{-1} * 2^{-5} * 2^{-1} = 2^{-7} = 0,0078125$ | $2^{-1} * 2^{-9} * 2^{-1} = 2^{-11} = 0,00048828125$ |
| 2,5    | $2^{-1} * 2^{-5} * 2^2 = 2^{-4} = 0,0625$       | $2^{-1} * 2^{-9} * 2^2 = 2^{-8} = 0,00390625$        |

**Ejercicio 10.**

Tomar un sistema de punto flotante cualquiera y dibujar la forma del gráfico de cada tipo de error en función del número que se quiere representar.

Supongo mantisa fraccionaria en BSS de 3 bits y exponente en BCS de 3 bits.



**Ejercicio 11.**

*Detallar las características del estándar IEEE 754 para simple precisión y doble precisión.*

| Característica      | Simple precisión      | Doble precisión         |
|---------------------|-----------------------|-------------------------|
| Bits en signo       | 1                     | 1                       |
| Bits en exponente   | 8                     | 11                      |
| Bits en fracción    | 23                    | 52                      |
| Total de bits       | 32                    | 64                      |
| Exponente en exceso | 127                   | 1023                    |
| Rango de exponente  | $[-126; 127]$         | $[-1022; 1023]$         |
| Rango de números    | $[2^{-126}; 2^{128}]$ | $[2^{-1022}; 2^{1024}]$ |

*¿Qué valores están representados por las siguientes cadenas si responden al estándar IEEE 754?*

$$\begin{aligned} 0\ 11000100\ [1]000000000000000000000000 &= 2^{69} * 2^0 \\ 0\ 11000100\ [1]000000000000000000000000 &= 2^{69} * 1 \\ 0\ 11000100\ [1]000000000000000000000000 &= 2^{69}. \end{aligned}$$
$$\begin{aligned} &1\ 11111110\ [1]10100000000000000000000 = -2^{127}\ (2^0 + 2^{-1} + 2^{-3}) \\ &1\ 11111110\ [1]10100000000000000000000 = -2^{127}\ (1 + 0,5 + 0,125) \\ &1\ 11111110\ [1]10100000000000000000000 = -1,625 * 2^{127}. \end{aligned}$$
[illegible]
$$\begin{aligned} 0\ 00000000[1]\ 100110000000000000000000 &= 2^{-126} (2^{-1} + 2^{-4} + 2^{-5}) \\ 0\ 00000000[1]\ 100110000000000000000000 &= 2^{-127} + 2^{-130} + 2^{-131}. \end{aligned}$$
$$1\ 00000000\ 000000000000000000000000=0.$$
$$0\ 11111111\ 00000000000000000000=+\infty.$$

(g)

**(h)**

(i)

(j)

$$(\mathbf{k})$$

```
1 1111111111 11111000000000000000000000000000000000000000= NaN.
```

**Ejercicio 13.**

Hallar la representación en simple precisión del estándar IEEE 754 de los siguientes números: 1, 13, 257, -40000, 0.0625.

| Número | Simple precisión del estándar IEEE 754    |
|--------|---|
| 1      | 0 01111111<br>[1]000000000000000000000000 |
| 13     | 0 10000010<br>[1]101000000000000000000000 |
| 257    | 0 10000111<br>[1]000000001000000000000000 |
| -40000 | 1 10001110<br>[1]001110001000000000000000 |
| 0,0625 | 0 01111011<br>[1]000000000000000000000000 |



**Ejercicio 14.**

Calcular rango y resolución en extremos inferior negativo y superior positivo para los sistemas de simple precisión y doble precisión del estándar IEEE 754. ¿Cuál es el menor número positivo distinto de "0" que se puede representar?

| Característica                       | Simple precisión                                  | Doble precisión                                     |
|--------------------------------------|---|---|
| Rango negativo                       | $[-(2 - 2^{-23}) * 2^{127}; -2^{-23} * 2^{-126}]$ | $[-(2 - 2^{-52}) * 2^{1023}; -2^{-52} * 2^{-1022}]$ |
| Rango positivo                       | $[2^{-23} * 2^{-126}; (2 - 2^{-23}) * 2^{127}]$   | $[2^{-52} * 2^{-1022}; (2 - 2^{-52}) * 2^{1023}]$   |
| Resolución extremo inferior negativo | $2^{-23} * 2^{127}$                               | $2^{-52} * 2^{1023}$                                |
| Resolución extremo superior positivo | $2^{-23} * 2^{127}$                               | $2^{-52} * 2^{1023}$                                |
| Menor número positivo distinto de 0  | $2^{-126} * 2^{-23}$                              | $2^{-1022} * 2^{-52}$                               |

**Ejercicio 15.**

Efectuar las siguientes sumas (las cadenas son representaciones en el estándar IEEE 754):

(a)  $00001111\ 010000000000000000000000 + 00010000\ 010000000000000000000000$ .

$$00001111\ [1],010000000000000000000000 + \\ 00010000\ [1],010000000000000000000000 =$$

$$2^{-112}_{(10)} * [1],010000000000000000000000 + \\ 2^{-111}_{(10)} * [1],010000000000000000000000 =$$

$$2^{-112}_{(10)} * [1],010000000000000000000000 + \\ 2^{-112}_{(10)} * [1]\ 0,100000000000000000000000 =$$

$$2^{-112}_{(10)} * [1]1,110000000000000000000000 =$$

$$2^{-111}_{(10)} * [1],111000000000000000000000 = 1,875 * 2^{-111}.$$

(b)  $11111111\ 101010101010101010101010 + 11111100\ 100000011111000001101010$ .

$$11111111\ [1]101010101010101010101010 + \\ 11111100\ [1]100000011111000001101010 =$$

$$\text{NaN} + \\ 2^{125}_{(10)} * [1]100000011111000001101010 = \\ \text{NaN}.$$

### **Ejercicio 16.**

*En el estándar IEEE 754, ¿para qué sirve, cuando el exponente es 0 y la mantisa no es nula, que la mantisa no esté normalizada?*

Cuando el exponente es 0 y la mantisa no es nula, que la mantisa no esté normalizada (desnormalizar) sirve para representar números por debajo de  $2^{2^0 - \text{exceso}}$  y para garantizar la menor brecha entre el menor número normalizado y el mayor número desnormalizado.

## Práctica 3

### Lógica y compuertas (Parte 1): Funciones lógicas elementales. Puertas lógicas.

*Objetivos de la práctica: que el alumno sea capaz de:*

- Realizar operaciones lógicas
- Usar máscaras y realizar equivalencias entre operaciones sucesivas.
- Establecer la salida de circuitos combinatorios simples.
- Confeccionar tablas de verdad.
- Describir la relación entre entradas y salidas por ecuaciones.

*Bibliografía:*

- “Organización y Arquitectura de Computadoras” de W. Stallings, Apéndice A, pág. 645.
- “Principios de Arquitectura de Computadoras” de Miles J. Murdocca, apéndice A, pág. 441.
- Apunte 3 de la cátedra, “Sistemas de Numeración: Operaciones Lógicas”.

### Operaciones Lógicas

1. Realizar las siguientes operaciones lógicas:

- 10011001 **AND** 10101110
- 01011000 **AND** 11110011
- 10011001 **OR** 10101110
- 01011000 **OR** 11110011
- 10011001 **XOR** 10101110
- 01011000 **XOR** 11110011
- NOT** 01011000
- NOT** 111010100
- 10011001 **NAND** 10101110
- 01011000 **NAND** 11110011
- 10111001 **NOR** 11101110
- 01011010 **NOR** 11010011
- 10111001 **XNOR** 11101110
- 01011010 **XNOR** 01011010

2. Dado un byte  $X=[X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0]$  (los X representan bits con valores indeterminados), ¿qué resultado obtendré al aplicarle una operación lógica junto a un valor predeterminado (máscara)? Analice para cada operación cómo los bits de la ‘máscara’ condicionan el resultado que se obtendrá. **¿Puede reconocer un patrón para cada máscara?**

En los casos de más de una operación, obtenga el resultado y a ese resultado aplíquelo la operación siguiente. Ejemplo:

|     |  |
|-----|--|
|     | $X_7X_6X_5X_4X_3X_2X_1X_0$   |
| AND | 0 <sub>7</sub> 1 <sub>6</sub> 0 <sub>5</sub> 0 <sub>4</sub> 0 <sub>3</sub> 1 <sub>2</sub> 0 <sub>1</sub> 0 <sub>0</sub>  |
|     | -----  |
|     | 0 <sub>7</sub> X <sub>6</sub> 0 <sub>5</sub> 0 <sub>4</sub> 0 <sub>3</sub> X <sub>2</sub> 0 <sub>1</sub> 0 <sub>0</sub>  |
| XOR | 0 <sub>7</sub> 1 <sub>6</sub> 0 <sub>5</sub> 0 <sub>4</sub> 0 <sub>3</sub> 0 <sub>2</sub> 1 <sub>1</sub> 0 <sub>0</sub>  |
|     | -----  |
|     | 0 <sub>7</sub> ~X <sub>6</sub> 0 <sub>5</sub> 0 <sub>4</sub> 0 <sub>3</sub> X <sub>2</sub> 1 <sub>1</sub> 0 <sub>0</sub> |

*Nota: ~X = X negado (valor opuesto a X)*

- X OR 00011000**
- X OR 11001100**
- X AND 01010101**
- X AND 01001100**
- X XOR 01010101**
- X XOR 11001100**

## Organización de Computadoras 2023

- g. **X OR 10000001**, al resultado **AND 00111001**, y al resultado **XOR 11001111**
- h. **X AND 10001110**, al resultado **OR 11001100**, y al resultado **XOR 01010011**
- i. **X XOR 10010010**, al resultado **AND 11100110**, y al resultado **OR 00110111**
- j. **X XNOR 10011001**, al resultado **NAND 11001100**, y al resultado **NOR 00011000**
- k. **X XOR 10100101**, al resultado **NAND 11100111**, y al resultado **NOR 01010110**

3. Complete las siguientes líneas punteadas con el operador lógico adecuado (sean AND, OR, XOR, NOT), en las siguientes expresiones de modo tal que se cumpla la igualdad propuesta:

- a. **1000 ..... 1101 = 1101**
- b. **1111 ..... 0101 = 0101**
- c. **1101 ..... 1001 = 0100**
- d. **..... (1111 ..... 0011) = 1100**
- e.  **$X_3 X_2 X_1 X_0$  ..... 1110 ..... 0101 ..... 0101 =  $X_3 0 X_1 0$**
- f.  **$X_3 X_2 X_1 X_0$  ..... 1000 ..... 1011 ..... 1110 =  $0 1 \underline{X_1} \underline{X_0}$**
- g. **.....( $X_3 X_2 X_1 X_0$  ..... 1001) =  $\underline{X_3} 11 \underline{X_0}$**

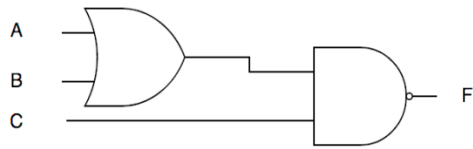
Se entiende que cada X es un bit desconocido que puede ser 1 o 0, debiendo obtenerse el resultado final al combinar diferentes operaciones lógicas, siguiendo el orden correcto.

4. Dado un byte **X=[X<sub>7</sub>,X<sub>6</sub>,X<sub>5</sub>,X<sub>4</sub>,X<sub>3</sub>,X<sub>2</sub>,X<sub>1</sub>,X<sub>0</sub>]** (los X representan bits con valores indeterminados), aplíquelo operaciones lógicas (1 o más) con un byte MASK, que deberá también determinar, para lograr los siguientes efectos:
- a) Poner en 1 los bits 1,3 y 5 dejando los demás bits iguales.
  - b) Poner a 1 los bits 4 y 6 dejando los demás iguales.
  - c) Poner a 0 los bits 1, 3 y 5 dejando los demás iguales.
  - d) Poner a 0 los bits 4 y 6 dejando los demás iguales.
  - e) Cambiar los bits 1, 3 y 5 a su complemento dejando los demás iguales.
  - f) Cambiar los bits 4 y 6 a su complemento dejando los demás iguales.
  - g) Poner en 1 los bits 1 y 5, poner en 0 los bits 7 y 0, cambiar el bit 6 por su complemento y dejar los demás iguales.
  - h) Poner en 0 los bits 1, 5 y 6, cambiar el bit 4 por su complemento y dejar los demás iguales.

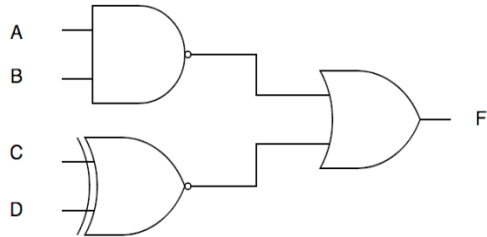
## Circuitos Combinatorios

3. Construir la tabla de verdad de los siguientes circuitos. Especifique además la ecuación que describe la relación entre entradas-salidas.

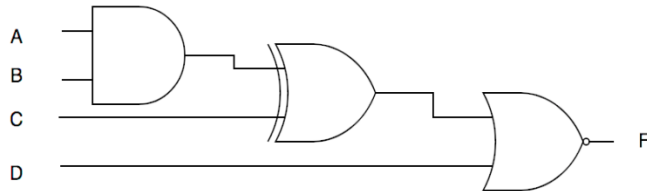
1)



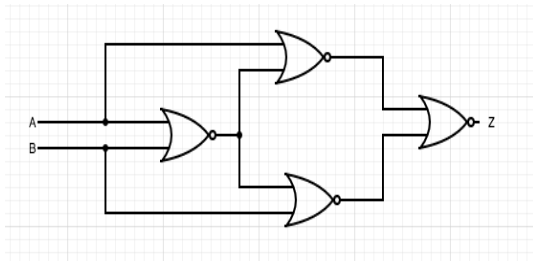
2)



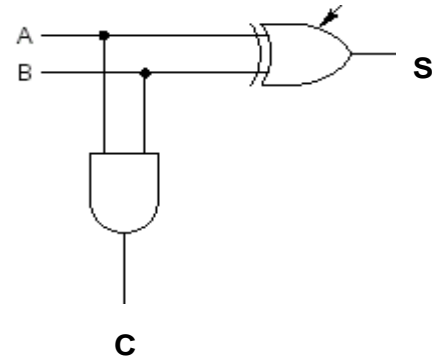
3)



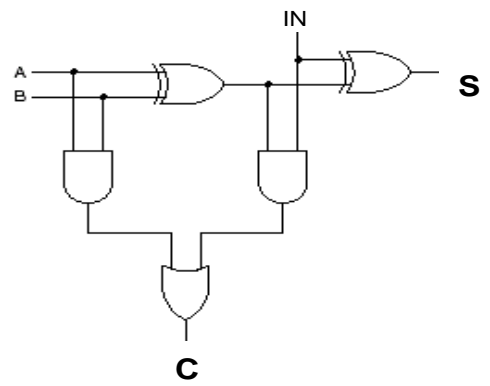
6)



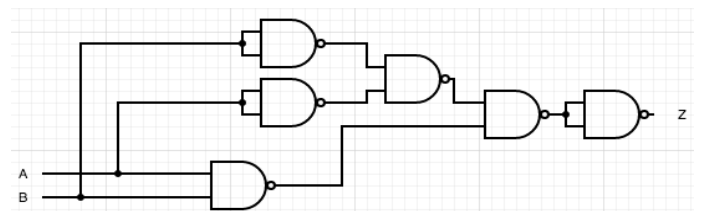
4)



5)



7)



## Lógica y compuertas (Parte 2): Circuitos Combinacionales y Secuenciales.

**Objetivos de la práctica: que el alumno domine**

- Circuitos lógicos y diagramas de compuertas
- Introducción a equivalencias lógicas
- método de sumas de productos.
- Describir el funcionamiento de los distintos tipos de flip flops.
- Comprender el funcionamiento de un circuito secuencial.

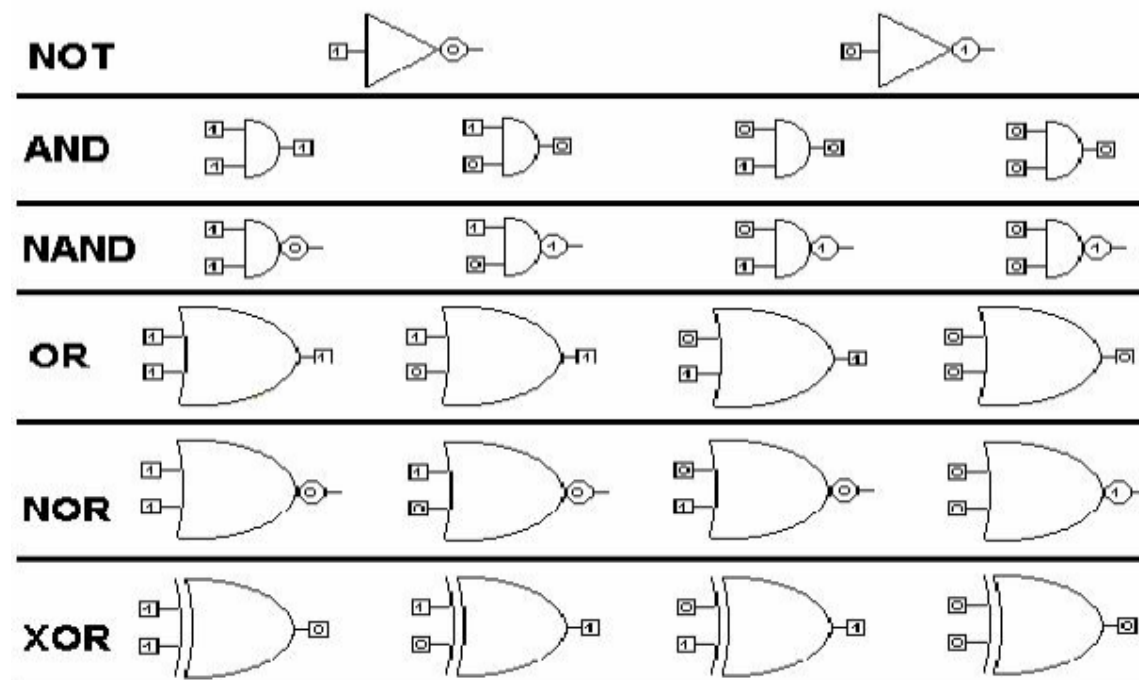
**Bibliografía:**

- "Principios de Arquitectura de Computadoras" de Miles J. Murdocca, apéndice A, pág. 441.
- Apunte 3 de la cátedra, "Sistemas de Numeración: Operaciones Lógicas".
- Apunte 5 de la cátedra, "Circuitos Lógicos Secuenciales".

Tener en cuenta para resolución de ejercicios 6 al 10:

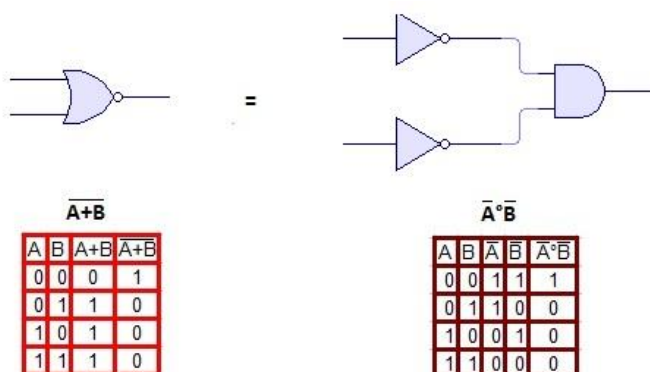
**Tablas de Verdad:** Una tabla de verdad muestra el resultado de una proposición compuesta para cada combinación de valores de verdad que se le puedan asignar a sus componentes de entrada.

Tengamos en cuenta las respuestas de los distintos conectivos lógicos/Compuertas:



**Equivalencias lógicas mediante tablas de verdad:** Es posible demostrar que dos circuitos son equivalentes si ante iguales entradas responden con el mismo valor de salida. Para llevar a cabo esta demostración, alcanza con construir la tabla de verdad de ambos circuitos y validar que las respuestas coinciden para iguales entradas.

Ejemplo: (La conocida Ley de De Morgan, donde se puede verificar que ante iguales combinaciones de valores de entrada para A y B, la respuesta del circuito es igual en ambos casos)



# Organización de Computadoras 2023

## Otras equivalencias lógicas:

### Conjunto cerrado de operaciones lógicas usando sólo compuertas Nand o Nor:

Es posible (su justificación excede el objetivo de este curso) reescribir cualquier expresión lógica compuesta, como una expresión equivalente utilizando EXCLUSIVAMENTE compuertas Nand o Nor. Esto favorece el diseño de circuitos al resolver cualquier lógica con un único tipo de compuertas.

### Equivalencias lógicas para representar cualquier conectivo lógico como compuertas Nand:

- $\overline{A} \cong \overline{A + A} \cong \overline{A.A}$  (Aplico 2 equivalencias lógicas, la última es la ley de De Morgan).
- $A + B \cong \overline{\overline{A + B}} \cong \overline{\overline{A}.B} \cong \overline{(\overline{A.A})(\overline{B.B})}$  (doble negación, De Morgan, equivalencia anterior para la negación).
- $A.B \cong \overline{\overline{A.B}} \cong \overline{(\overline{A.B})(\overline{A.B})}$  (doble negación, 1er equivalencia para la negación).
- $A \otimes B \cong (\overline{A.B}) + (\overline{A.B})$ .... (definición del or exclusivo, resta aplicar las equivalencias previas para producto, suma y negación para llegar a utilizar sólo compuertas Nand).

El resto de las compuertas pueden reescribirse sólo con compuertas Nand basándose en las equivalencias previas.

---

### 6. Demostrar mediante tabla de verdad si se cumplen o no las siguientes equivalencias:

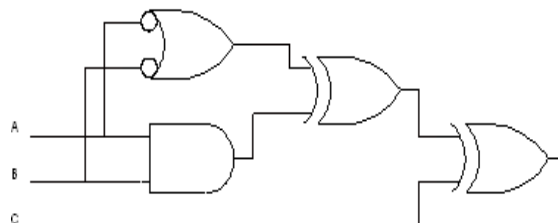
- $\overline{(A.B)} = \overline{A} + \overline{B}$  (La segunda ley de De Morgan)
- $A + B.C = (A + B) + (A + C)$
- $(A + B).C = (A.B) + (A.C)$
- $A + A + B = A + B + B$
- $A + B.C = A.C + B$
- $A \oplus B = A \oplus B$

### 7. Modifique los siguientes circuitos para que sean todas compuertas **NAND**.



### 8. Reescriba las compuertas lógicas Not, Or, And y Xor utilizando exclusivamente compuertas **NOR**. (Ver como se resolvió el mismo caso para compuertas Nand, en *Tener en . . .*).

### 9. Construya la tabla de verdad del siguiente circuito. Analice los valores y basándose en sus conclusiones construya un diagrama más simple que implemente la misma función de salida. Escriba además la ecuación de salida en forma de función.





# Organización de Computadoras 2023

10. Dadas las siguientes relaciones, dibuje los diagramas de compuertas que cumplen con ellas. Modifíquelos utilizando sólo compuertas **NOR**. Modifíquelos utilizando sólo compuertas **NAND**.

a)  $F = AB + AC + AD + \overline{ABCD}$

b)  $F = \overline{A + B + C + D}$

c)  $F = \overline{A + B\overline{C} + C}$

d)  $F = \overline{AB} + \overline{AB}$

Tener en cuenta para ejercicios 11 al 13:

**Suma de Productos:** Es posible inferir la fórmula lógica asociada a una función desconocida de la cual sólo se conoce la respuesta ante todas las combinaciones posibles de entradas....

Ejemplo: Supongamos una función que recibe 2 parámetros A y B, si conocemos la respuesta F de la ecuación en base a los posibles valores de A y B mediante la siguiente tabla de verdad:

| A | B | F |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

¿En qué casos la salida F será 1? Rta: Cuando las entradas sean A=0 y B=0, o A=0 y B=1, o A=1 y B=1.

Dicho de otra manera, podemos interpretar como respuesta válida que F será 1 cuando no ocurra A y no ocurra B, o no ocurra A y sí ocurra B, o cuando ocurran A y B.

Esto que es tan simple de entender en lenguaje cotidiano, se traslada con el mismo concepto a la idea de suma de productos, considerando que estamos haciendo una Disyunción/Suma (con la simbología que deseemos: O, Or, v, +) de Conjunciones/Productos (simbología: y, And, ^, .). En conclusión podemos inferir de la anterior tabla de verdad lo siguiente:

$$F = \overline{A}.\overline{B} + \overline{A}.B + A.B$$

(Por convención y de manera análoga a las operaciones aritméticas conocidas entendemos que ante la ausencia de paréntesis se calculan primero los productos y luego las sumas con los resultados intermedios de cada producto).

Para validar la veracidad de lo expuesto, se debe armar la tabla de verdad de la proposición compuesta y comprobar que coinciden las salidas para todas las combinaciones posibles de la tabla original.

Imaginemos ahora una función que recibe 4 variables A,B,C,D que representan los 4 dígitos de un número binario (Siendo D el menos significativo hasta A como más significativo)....Respondamos ahora la siguiente pregunta:

¿Cuándo viene representado el número 5? (Sabemos que el 5 se representa en binario como 0101)

Rta: cuando viene A=0 y B=1 y C=0 y D=1. O dicho de otra manera, cuando NO ocurra A y SI ocurra B y NO ocurra C y SI ocurra D.

Conclusión: Se puede representar una ecuación que retorne 1 cuando en las cuatro entradas reciba el número 5, de la siguiente manera:  $F_5 = \overline{A}.B.\overline{C}.D$  (Notar que la salida  $F_5$  tomará valor 1 exclusivamente cuando las entradas ABCD sean 0101)

Ahora estamos preparados para determinar una ecuación que, por ejemplo, retorne 1 cuando el número representado en las cuatro entradas sea 5 o 7 o 9 (es decir 0101 o 0111 o 1001)

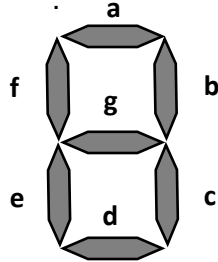
$$F = \overline{A}.B.\overline{C}.D + \overline{A}.B.C.D + A.\overline{B}.\overline{C}.D$$

(Notar que la salida F tomará el valor 1 exclusivamente cuando las entradas sean alguna de las 3 definidas, en cualquier otra combinación de entrada, la ecuación retornará 0).

11. Para la siguiente tabla de verdad encuentre una fórmula lógica correspondiente (utilizando suma de productos).

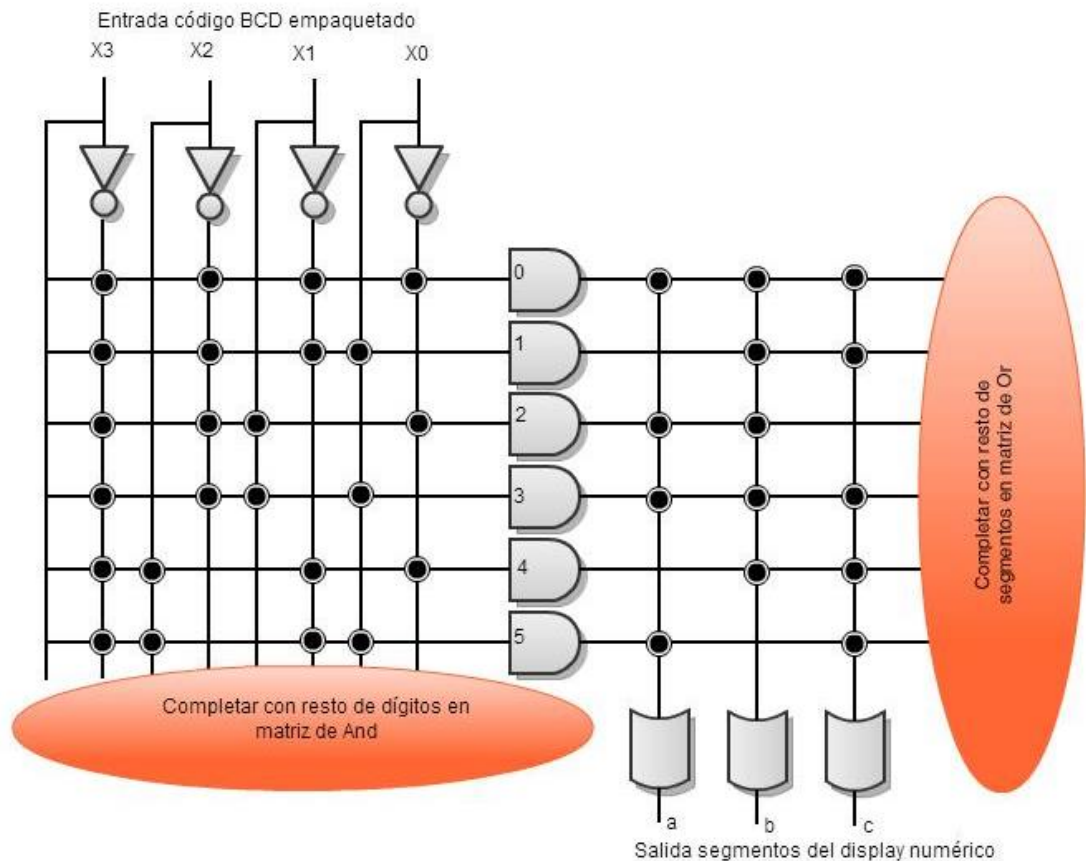
| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

12. Diseñe un circuito que tenga como entrada código BCD empaquetado (4 entradas) y 7 salidas para controlar los 7 segmentos de un display numérico, siendo la salida para los segmentos '0' para apagado y '1' para prendido. Construya la tabla de verdad y la ecuación de la salida correspondiente a los segmentos **a**, **b**, **c**, **d**, **e**, **f** y **g**.



**Ayuda 1:** Cada segmento se considera como una salida distinta, y cada uno se debe activar (poner en 1) dependiendo del número recibido en las entradas que representan los 4 bits de un BCD empaquetado. Ejemplo: El segmento **b** se debe activar cuando se recibe un 1 (0001), o un 2 (0010), o un 3 (00110), o un 4 (0100), o un 7 (0111), o un 8 (1000), o un **9 (1001)**. **Se aplica la misma idea con el resto de las salidas.**

**Ayuda 2:** Gráficamente, el circuito con las 4 entradas y las 7 salidas conviene diseñarlo como una matriz de compuertas And, seguida de la matriz de compuertas Or (basarse en la siguiente gráfica parcial



13. Un controlador de proceso industrial recibe como entrada tres señales de temperatura T1, T2, T3 ( $T1 < T2 < T3$ ) que adoptan el valor lógico '1' cuando la temperatura es mayor que t1, t2 y t3 respectivamente. Diseñar un circuito que genere una señal F cuando la temperatura esté comprendida entre t1 y t2 o cuando la temperatura sea mayor que t3.

# Organización de Computadoras 2023

Tener en cuenta para ejercicios 14 al 18:

**Circuitos Secuenciales:** (repasar apuntes de la cátedra y teoría)

- Flip flop S-R asincrónico:
  - Problemas de sincronismo ante cambios de entrada durante el cálculo.
  - Reacción frente a doble entrada de 1's.
- Flip flop S-R sincrónico:
  - Resuelve problema de sincronismo, pero mantiene problema ante doble entrada de 1's.
- Flip flop D:
  - Pequeña variante del S-R que resuelve el problema de la doble entrada de 1's.
- Flip flop J-K:
  - Incorpora posibilidad de alterar el valor previo (complemento lógico).
- Flip flop T:
  - Pequeña variante del J-K, que sólo se dedica a invertir su valor ante cada orden del clock.

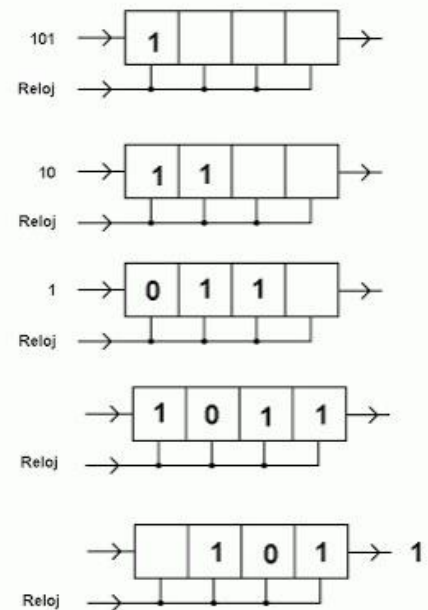
**14.** Dibuje el esquema de compuertas que componen un flip flop S-R. Describa a través de una tabla los estados en función de las entradas. Modifique el esquema anterior para hacerlo sincrónico. Describa gráficamente su respuesta temporal.

**15.** Dibuje el esquema de un flip flop D. Detalle en su respuesta temporal como resuelve el problema de la doble entrada de 1's que se presentaba en el S-R.

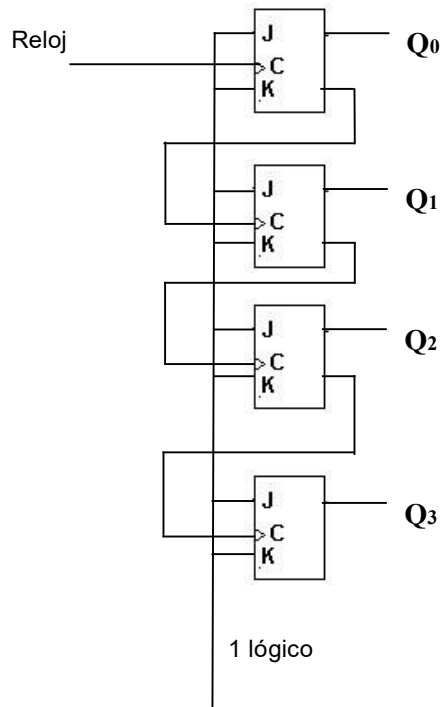
**16.** Dibuje el esquema de un flip flop J-K, describiendo su respuesta temporal.

**17.** Dibuje el diagrama de tiempos del registro de la figura, implementado con flip flops D. Modifíquelo para desplazamiento izquierda derecha y derecha izquierda.

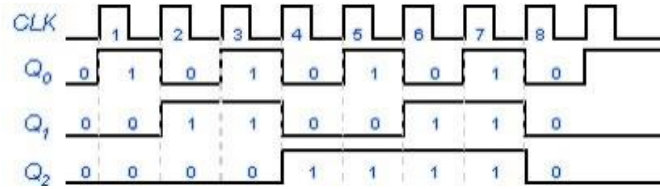
Ayuda: Ejemplo de respuesta temporal para interpretar como responde el registro previo ante la entrada serial del número binario 1011:



18. Describa gráficamente la respuesta temporal de cada flip flop ante una señal de unos y ceros entrando por Reloj.



Ayuda: El diagrama correspondiente considerando sólo los primeros 3 flip-Flops es el siguiente:



Se observa como la respuesta de cada flip-flop emite una onda a la mitad de frecuencia que su clock de entrada.

## **Trabajo Práctico N° 3:** **Lógica y Puertas.**

### **FUNCIONES LÓGICAS ELEMENTALES. PUERTAS LÓGICAS**

#### **Ejercicio 1.**

*Realizar las siguientes operaciones lógicas:*

| <b>Inciso</b> | <b>Operación</b> | <b>Cadena 1</b> | <b>Cadena 2</b> | <b>Resultado</b> |
|---------------|------------------|-----------------|-----------------|------------------|
| (a)           | AND              | 10011001        | 10101110        | 10001000         |
| (b)           | AND              | 01011000        | 11110011        | 01010000         |
| (c)           | OR               | 10011001        | 10101110        | 10111111         |
| (d)           | OR               | 01011000        | 11110011        | 11111011         |
| (e)           | XOR              | 10011001        | 10101110        | 00110111         |
| (f)           | XOR              | 01011000        | 11110011        | 10101011         |
| (g)           | NOT              | 010111000       | ---             | 101000111        |
| (h)           | NOT              | 111010100       | ---             | 000101011        |
| (i)           | NAND             | 10011001        | 10101110        | 01110111         |
| (j)           | NAND             | 01011000        | 11110011        | 10101111         |
| (k)           | NOR              | 10111001        | 11101110        | 00000000         |
| (l)           | NOR              | 01011010        | 11010011        | 00100100         |
| (m)           | XNOR             | 10111001        | 11101110        | 10101000         |
| (n)           | XNOR             | 01011010        | 01011010        | 11111111         |

**Ejercicio 2.**

Dado un byte  $X = [X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0]$  (los  $X$  representan bits con valores indeterminados), ¿qué resultado se obtendrá al aplicarle una operación lógica junto a un valor predeterminado (máscara)? Analizar, para cada operación, cómo los bits de la “máscara” condicionan el resultado que se obtendrá. ¿Puede reconocer un patrón para cada máscara? En los casos de más de una operación, obtener el resultado y a ese resultado aplicarle la operación siguiente.

(a)  $X \text{ OR } 00011000$ .

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{OR} \quad 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \\
 \hline
 X_7 X_6 X_5 1 \ 1 \ X_2 X_1 X_0
 \end{array}$$

(b)  $X \text{ OR } 11001100$ .

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{OR} \quad 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \\
 \hline
 1 \ 1 \ X_5 X_4 \ 1 \ 1 \ X_1 X_0
 \end{array}$$

(c)  $X \text{ AND } 01010101$ .

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{AND} \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \\
 \hline
 0 \ X_6 0 \ X_4 0 \ X_2 0 \ X_0
 \end{array}$$

(d)  $X \text{ AND } 01001100$ .

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{AND} \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \\
 \hline
 0 \ X_6 0 \ 0 \ X_3 X_2 0 \ 0
 \end{array}$$

(e)  $X \text{ XOR } 01010101$ .

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{XOR} \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1
 \end{array}$$

$$X_7 \bar{X}_6 X_5 \bar{X}_4 X_3 \bar{X}_2 X_1 \bar{X}_0$$

(f)  $X \text{ XOR } 11001100$ .

$$\begin{array}{r} X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\ \text{XOR } 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \end{array}$$

$$\bar{X}_7 \bar{X}_6 X_5 X_4 \bar{X}_3 \bar{X}_2 X_1 X_0$$

(g)  $X \text{ OR } 10000001 \text{ AND } 00111001 \text{ XOR } 11001111$ .

$$\begin{array}{r} X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\ \text{OR } 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \end{array}$$

$$\begin{array}{r} 1 \ X_6 X_5 X_4 X_3 X_2 X_1 1 \\ \text{AND } 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \end{array}$$

$$\begin{array}{r} 0 \ 0 \ X_5 X_4 X_3 0 \ 0 \ 1 \\ \text{XOR } 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \end{array}$$

$$1 \ 1 \ X_5 X_4 \bar{X}_3 1 \ 1 \ 0$$

(h)  $X \text{ AND } 10001110 \text{ OR } 11001100 \text{ XOR } 01010011$ .

$$\begin{array}{r} X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\ \text{AND } 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \end{array}$$

$$\begin{array}{r} X_7 0 \ 0 \ 0 \ X_3 X_2 X_1 0 \\ \text{OR } 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \end{array}$$

$$\begin{array}{r} 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ X_1 0 \\ \text{XOR } 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \end{array}$$

$$1 \ 0 \ 0 \ 1 \ 1 \ 1 \ \bar{X}_1 1$$

(i)  $X \text{ XOR } 10010010 \text{ AND } 11100110 \text{ OR } 00110111$ .

$$\begin{array}{r}
 X_7X_6X_5X_4X_3X_2X_1X_0 \\
 \text{XOR } 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \\
 \hline
 \bar{X}_7X_6X_5\bar{X}_4X_3X_2\bar{X}_1X_0 \\
 \text{AND } 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \\
 \hline
 \bar{X}_7X_6X_50 \ 0 \ X_2\bar{X}_10 \\
 \text{OR } 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \\
 \hline
 \bar{X}_7X_61 \ 1 \ 0 \ 1 \ 1 \ 1
 \end{array}$$

(j)  $X \text{ XNOR } 10011001 \text{ NAND } 11001100 \text{ NOR } 00011000$ .

$$\begin{array}{r}
 X_7X_6X_5X_4X_3X_2X_1X_0 \\
 \text{XNOR } 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \\
 \hline
 X_7\bar{X}_6\bar{X}_5X_4X_3\bar{X}_2\bar{X}_1X_0 \\
 \text{NAND } 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \\
 \hline
 \bar{X}_7X_61 \ 1 \ \bar{X}_3X_21 \ 1 \\
 \text{NOR } 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \\
 \hline
 X_7\bar{X}_60 \ 0 \ 0 \ \bar{X}_20 \ 0
 \end{array}$$

(k)  $X \text{ XOR } 10100101 \text{ NAND } 11100111 \text{ NOR } 01010110$ .

$$\begin{array}{r}
 X_7X_6X_5X_4X_3X_2X_1X_0 \\
 \text{XOR } 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \\
 \hline
 \bar{X}_7X_6\bar{X}_5X_4X_3\bar{X}_2X_1\bar{X}_0 \\
 \text{NAND } 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \\
 \hline
 X_7\bar{X}_6X_51 \ 1 \ X_2\bar{X}_1X_0 \\
 \text{NOR } 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \\
 \hline
 \bar{X}_70 \ \bar{X}_50 \ 0 \ 0 \ 0 \ 0
 \end{array}$$



**Ejercicio 3.**

Completar las siguientes líneas punteadas con el operador lógico adecuado (sean AND, OR, XOR, NOT) en las siguientes expresiones de modo tal que se cumpla la igualdad propuesta. Se entiende que cada  $X$  es un bit desconocido que puede ser 1 o 0, debiendo obtenerse el resultado final al combinar diferentes operaciones lógicas, siguiendo el orden correcto.

(a)

$$1000 \text{ OR } 1101 = 1101.$$

(b)

$$1111 \text{ AND } 0101 = 0101.$$

(c)

$$1101 \text{ XOR } 1001 = 0100.$$

(d)

$$\text{NOT } (1111 \text{ XOR } 0011) = 1100.$$

$$\text{NOT } 0011 = 1100.$$

(e)

$$X_3X_2X_1X_0 \text{ AND } 1110 \text{ OR } 0101 \text{ XOR } 0101 = X_30X_10.$$

$$X_3X_2X_10 \text{ OR } 0101 \text{ XOR } 0101 = X_30X_10.$$

$$X_31X_11 \text{ XOR } 0101 = X_30X_10.$$

(f)

$$X_3X_2X_1X_0 \text{ OR } 1000 \text{ AND } 1011 \text{ OR } 1110 = 01\bar{X}_1X_0.$$

$$1X_2X_1X_0 \text{ AND } 1011 \text{ OR } 1110 = 01\bar{X}_1X_0.$$

$$10X_1X_0 \text{ XOR } 1110 = 01\bar{X}_1X_0.$$

(g)

**NOT**  $(X_3X_2X_1X_0 \text{ AND } 1001) = \bar{X}_311\bar{X}_0$ .

**NOT**  $X_300X_0 = \bar{X}_311\bar{X}_0$ .

**Ejercicio 4.**

Dado un byte  $X = [X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0]$  (los  $X$  representan bits con valores indeterminados), aplicar operaciones lógicas (1 o más) con un byte MASK, que se deberá también determinar, para lograr los efectos:

(a) Poner en 1 los bits 1,3 y 5 dejando los demás bits iguales.

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{OR} \quad 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \\
 \hline
 X_7 X_6 1 \ X_4 1 \ X_2 1 \ X_0
 \end{array}$$

(b) Poner en 1 los bits 4 y 6 dejando los demás iguales.

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{OR} \quad 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 X_7 1 \ X_5 1 \ X_3 X_2 X_1 X_0
 \end{array}$$

(c) Poner en 0 los bits 1, 3 y 5 dejando los demás iguales.

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{AND} \quad 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \\
 \hline
 X_7 X_6 0 \ X_4 0 \ X_2 0 \ X_0
 \end{array}$$

(d) Poner en 0 los bits 4 y 6 dejando los demás iguales.

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{AND} \quad 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \\
 \hline
 X_7 0 \ X_5 0 \ X_3 X_2 X_1 X_0
 \end{array}$$

(e) Cambiar los bits 1, 3 y 5 a su complemento dejando los demás iguales.

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{XOR} \quad 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \\
 \hline
 X_7 X_6 \bar{X}_5 X_4 \bar{X}_3 X_2 \bar{X}_1 X_0
 \end{array}$$

(f) Cambiar los bits 4 y 6 a su complemento dejando los demás iguales.

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{XOR } 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 X_7 \bar{X}_6 X_5 \bar{X}_4 X_3 X_2 X_1 X_0
 \end{array}$$

(g) Poner en 1 los bits 1 y 5, poner en 0 los bits 7 y 0, cambiar el bit 6 por su complemento y dejar los demás iguales.

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{OR } 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \\
 \hline
 X_7 X_6 1 \ X_4 X_3 X_2 1 \ X_0 \\
 \text{AND } 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \\
 \hline
 0 \ X_6 1 \ X_4 X_3 X_2 1 \ 0 \\
 \text{XOR } 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 0 \ \bar{X}_6 1 \ X_4 X_3 X_2 1 \ 0
 \end{array}$$

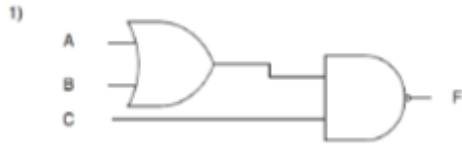
(h) Poner en 0 los bits 1, 5 y 6, cambiar el bit 4 por su complemento y dejar los demás iguales.

$$\begin{array}{r}
 X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 \\
 \text{AND } 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \\
 \hline
 X_7 0 \ 0 \ X_4 X_3 X_2 0 \ X_0 \\
 \text{XOR } 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \\
 \hline
 X_7 0 \ 0 \ \bar{X}_4 X_3 X_2 0 \ X_0
 \end{array}$$

**Ejercicio 5.**

Construir la tabla de verdad de los siguientes circuitos. Especificar, además, la ecuación que describe la relación entre entradas-salidas:

(a)

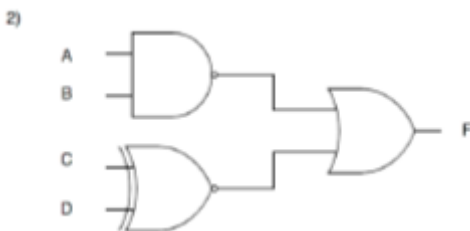


| A | B | C | A OR B | <b>F = (A OR B) AND C</b> |
|---|---|---|--------|---------------------------|
| 0 | 0 | 0 | 0      | 0                         |
| 0 | 0 | 1 | 0      | 0                         |
| 0 | 1 | 0 | 1      | 0                         |
| 0 | 1 | 1 | 1      | 1                         |
| 1 | 0 | 0 | 1      | 0                         |
| 1 | 0 | 1 | 1      | 1                         |
| 1 | 1 | 0 | 1      | 0                         |
| 1 | 1 | 1 | 1      | 1                         |

$$F = \bar{A}BC + A\bar{B}C + ABC.$$

$$F = (A + B + C)(A + B + \bar{C})(A + \bar{B} + C)(\bar{A} + B + C)(\bar{A} + \bar{B} + C).$$

(b)

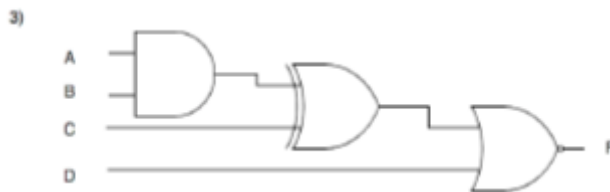


| A | B | C | D | A AND B | C XOR D | F= (A AND B) OR (C XOR D) |
|---|---|---|---|---------|---------|---------------------------|
| 0 | 0 | 0 | 0 | 0       | 0       | 0                         |
| 0 | 0 | 0 | 1 | 0       | 1       | 1                         |
| 0 | 0 | 1 | 0 | 0       | 1       | 1                         |
| 0 | 0 | 1 | 1 | 0       | 0       | 0                         |
| 0 | 1 | 0 | 0 | 0       | 0       | 0                         |
| 0 | 1 | 0 | 1 | 0       | 1       | 1                         |
| 0 | 1 | 1 | 0 | 0       | 1       | 1                         |
| 0 | 1 | 1 | 1 | 0       | 0       | 0                         |
| 1 | 0 | 0 | 0 | 0       | 0       | 0                         |
| 1 | 0 | 0 | 1 | 0       | 1       | 1                         |
| 1 | 0 | 1 | 0 | 0       | 1       | 1                         |
| 1 | 0 | 1 | 1 | 0       | 0       | 0                         |
| 1 | 1 | 0 | 0 | 1       | 0       | 1                         |
| 1 | 1 | 0 | 1 | 1       | 1       | 1                         |
| 1 | 1 | 1 | 0 | 1       | 1       | 1                         |
| 1 | 1 | 1 | 1 | 1       | 0       | 1                         |

$$F = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + ABC\bar{D} + ABC\bar{D} + ABCD.$$

$$F = (A + B + C + D) (A + B + \bar{C} + \bar{D}) (A + \bar{B} + C + D) (A + \bar{B} + \bar{C} + \bar{D}) (\bar{A} + B + C + D) (\bar{A} + B + \bar{C} + \bar{D}).$$

(c)



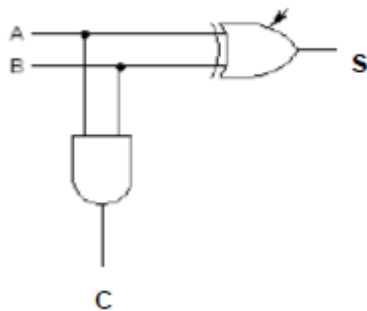
| A | B | C | D | A AND B | (A AND B) XOR C | F = [(A AND B) XOR C] OR D |
|---|---|---|---|---------|-----------------|----------------------------|
| 0 | 0 | 0 | 0 | 0       | 0               | 0                          |
| 0 | 0 | 0 | 1 | 0       | 0               | 1                          |
| 0 | 0 | 1 | 0 | 0       | 1               | 1                          |
| 0 | 0 | 1 | 1 | 0       | 1               | 1                          |
| 0 | 1 | 0 | 0 | 0       | 0               | 0                          |
| 0 | 1 | 0 | 1 | 0       | 0               | 1                          |
| 0 | 1 | 1 | 0 | 0       | 1               | 1                          |
| 0 | 1 | 1 | 1 | 0       | 1               | 1                          |
| 1 | 0 | 0 | 0 | 0       | 0               | 0                          |
| 1 | 0 | 0 | 1 | 0       | 0               | 1                          |
| 1 | 0 | 1 | 0 | 0       | 1               | 1                          |
| 1 | 0 | 1 | 1 | 0       | 1               | 1                          |
| 1 | 1 | 0 | 0 | 1       | 1               | 1                          |
| 1 | 1 | 0 | 1 | 1       | 1               | 1                          |
| 1 | 1 | 1 | 0 | 1       | 0               | 0                          |
| 1 | 1 | 1 | 1 | 1       | 0               | 1                          |

$$F = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}CD + AB\bar{C}\bar{D} + AB\bar{C}D + ABCD.$$

$$F = (A + B + C + D)(A + \bar{B} + C + D)(\bar{A} + B + C + D)(\bar{A} + \bar{B} + \bar{C} + D).$$

(d)

4)



| A | B | C = A AND B | S = A XOR B |
|---|---|-------------|-------------|
| 0 | 0 | 0           | 0           |
| 0 | 1 | 0           | 1           |
| 1 | 0 | 0           | 1           |
| 1 | 1 | 1           | 0           |

$$C = AB.$$

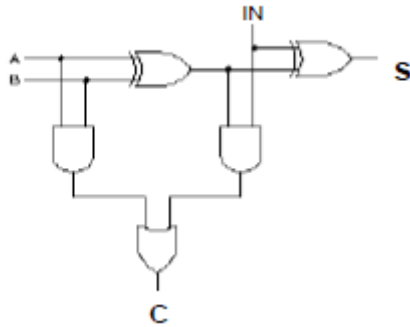
$$C = (A + B)(A + \bar{B})(\bar{A} + B).$$

$$S = \bar{A}B + A\bar{B}.$$

$$S = (A + B) (\bar{A} + \bar{B}).$$

(e)

5)



| A | B | IN | A AND B | A XOR B | (A XOR B) AND IN | C = (A AND B) OR [(A XOR B) AND IN] | S = (A XOR B) XOR IN |
|---|---|----|---------|---------|------------------|-------------------------------------|----------------------|
| 0 | 0 | 0  | 0       | 0       | 0                | 0                                   | 0                    |
| 0 | 0 | 1  | 0       | 0       | 0                | 0                                   | 1                    |
| 0 | 1 | 0  | 0       | 1       | 0                | 0                                   | 1                    |
| 0 | 1 | 1  | 0       | 1       | 1                | 1                                   | 0                    |
| 1 | 0 | 0  | 0       | 1       | 0                | 0                                   | 1                    |
| 1 | 0 | 1  | 0       | 1       | 1                | 1                                   | 0                    |
| 1 | 1 | 0  | 1       | 0       | 0                | 1                                   | 0                    |
| 1 | 1 | 1  | 1       | 0       | 0                | 1                                   | 1                    |

$$C = \bar{A}B\text{IN} + A\bar{B}\text{IN} + AB\bar{\text{IN}} + AB\text{IN}.$$

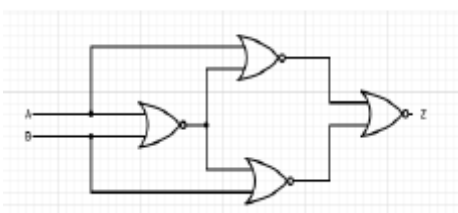
$$C = (A + B + \text{IN}) (A + B + \bar{\text{IN}}) (A + \bar{B} + \text{IN}) (\bar{A} + B + C).$$

$$S = \bar{A}\bar{B}\text{IN} + \bar{A}B\bar{\text{IN}} + A\bar{B}\bar{\text{IN}} + AB\text{IN}.$$

$$S = (A + B + \text{IN}) (A + \bar{B} + \bar{\text{IN}}) (\bar{A} + B + \bar{\text{IN}}) (\bar{A} + \bar{B} + \text{IN}).$$

(f)

6)





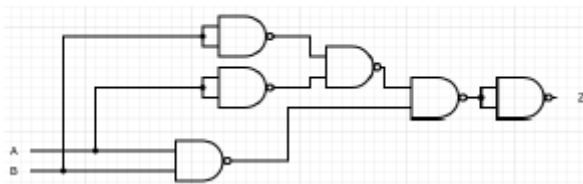
| A | B | A NOR B | A NOR (A NOR B) | B NOR (A NOR B) | Z = [A NOR (A NOR B)] NOR [B NOR (A NOR B)] |
|---|---|---------|-----------------|-----------------|---|
| 0 | 0 | 1       | 0               | 0               | 1   |
| 0 | 1 | 0       | 1               | 0               | 0   |
| 1 | 0 | 0       | 0               | 1               | 0   |
| 1 | 1 | 0       | 0               | 0               | 1   |

$$Z = \bar{A}\bar{B} + AB.$$

$$Z = (A + \bar{B})(\bar{A} + B).$$

(g)

7)



| A | B | A<br>NAND<br>B | A<br>NAND<br>A | B<br>NAND<br>B | (A<br>NAND<br>A)<br>NAND<br>(B<br>NAND<br>B) | (A<br>NAND<br>B)<br>NAND<br>[(A<br>NAND<br>A)<br>NAND<br>(B<br>NAND<br>B)] | Z= [(A<br>NAND<br>B)<br>NAND<br>[(A<br>NAND<br>A)<br>NAND<br>(B<br>NAND<br>B)]] |
|---|---|----------------|----------------|----------------|--|--|---|
| 0 | 0 | 1              | 1              | 1              | 0  | 1  | 0   |
| 0 | 1 | 1              | 1              | 0              | 1  | 0  | 1   |
| 1 | 0 | 1              | 0              | 1              | 1  | 0  | 1   |
| 1 | 1 | 0              | 0              | 0              | 1  | 1  | 0   |

$$Z = \bar{A}B + A\bar{B}.$$

$$Z = (A + B) (\bar{A} + \bar{B}).$$

**CIRCUITOS COMBINACIONALES Y SECUENCIALES****Ejercicio 6.**

*Demostrar mediante tabla de verdad si se cumplen o no las siguientes equivalencias:*

(a)  $\overline{AB} = \bar{A} + \bar{B}$ .

| A | B | $\bar{A}$ | $\bar{B}$ | AB | $\overline{AB}$ | $\bar{A} + \bar{B}$ |
|---|---|-----------|-----------|----|-----------------|---------------------|
| 0 | 0 | 1         | 1         | 0  | 1               | 1                   |
| 0 | 1 | 1         | 0         | 0  | 1               | 1                   |
| 1 | 0 | 0         | 1         | 0  | 1               | 1                   |
| 1 | 1 | 0         | 0         | 1  | 0               | 0                   |

Por lo tanto, sí se cumple esta equivalencia.

(b)  $A + BC = (A + B) + (A + C)$ .

| A | B | C | BC | A + B | A + C | A + BC | (A + B) + (A + C) |
|---|---|---|----|-------|-------|--------|-------------------|
| 0 | 0 | 0 | 0  | 0     | 0     | 0      | 0                 |
| 0 | 0 | 1 | 0  | 0     | 1     | 0      | 1                 |
| 0 | 1 | 0 | 0  | 1     | 0     | 0      | 1                 |
| 0 | 1 | 1 | 1  | 1     | 1     | 1      | 1                 |
| 1 | 0 | 0 | 0  | 1     | 1     | 1      | 1                 |
| 1 | 0 | 1 | 0  | 1     | 1     | 1      | 1                 |
| 1 | 1 | 0 | 0  | 1     | 1     | 1      | 1                 |
| 1 | 1 | 1 | 1  | 1     | 1     | 1      | 1                 |

Por lo tanto, no se cumple esta equivalencia.

(c)  $(A + B)C = AB + AC$ .

| A | B | C | A + B | AB | AC | (A + B)C | AB + AC |
|---|---|---|-------|----|----|----------|---------|
| 0 | 0 | 0 | 0     | 0  | 0  | 0        | 0       |
| 0 | 0 | 1 | 0     | 0  | 0  | 0        | 0       |
| 0 | 1 | 0 | 1     | 0  | 0  | 0        | 0       |
| 0 | 1 | 1 | 1     | 0  | 0  | 1        | 0       |
| 1 | 0 | 0 | 1     | 0  | 0  | 0        | 0       |
| 1 | 0 | 1 | 1     | 0  | 1  | 1        | 1       |
| 1 | 1 | 0 | 1     | 1  | 0  | 0        | 1       |
| 1 | 1 | 1 | 1     | 1  | 1  | 1        | 1       |

Por lo tanto, no se cumple esta equivalencia.

(d)  $A + A + B = A + B + B$ .

| A | B | $A + A + B$ | $A + B + B$ |
|---|---|-------------|-------------|
| 0 | 0 | 0           | 0           |
| 0 | 1 | 1           | 1           |
| 1 | 0 | 1           | 1           |
| 1 | 1 | 1           | 1           |

Por lo tanto, sí se cumple esta equivalencia.

(e)  $A + B\bar{C} = A\bar{C} + B$ .

| A | B | $\bar{C}$ | $B\bar{C}$ | $A\bar{C}$ | $A + B\bar{C}$ | $A\bar{C} + B$ |
|---|---|-----------|------------|------------|----------------|----------------|
| 0 | 0 | 0         | 0          | 0          | 0              | 0              |
| 0 | 0 | 1         | 0          | 0          | 0              | 0              |
| 0 | 1 | 0         | 0          | 0          | 0              | 1              |
| 0 | 1 | 1         | 1          | 0          | 1              | 1              |
| 1 | 0 | 0         | 0          | 0          | 1              | 0              |
| 1 | 0 | 1         | 0          | 1          | 1              | 1              |
| 1 | 1 | 0         | 0          | 0          | 1              | 1              |
| 1 | 1 | 1         | 1          | 1          | 1              | 1              |

Por lo tanto, no se cumple esta equivalencia.

(f)  $A \oplus B = A \oplus B$ .

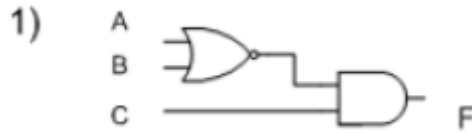
| A | B | $A \oplus B$ |
|---|---|--------------|
| 0 | 0 | 0            |
| 0 | 1 | 1            |
| 1 | 0 | 1            |
| 1 | 1 | 0            |

Por lo tanto, sí se cumple esta equivalencia.

**Ejercicio 7.**

Modificar los siguientes circuitos para que sean todas compuertas NAND.

(a)



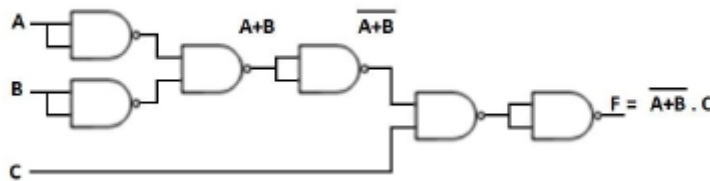
$$A \text{ NOR } B = \overline{A + B}.$$

$$(A \text{ NOR } B) \text{ AND } C = \overline{A + B} * C.$$

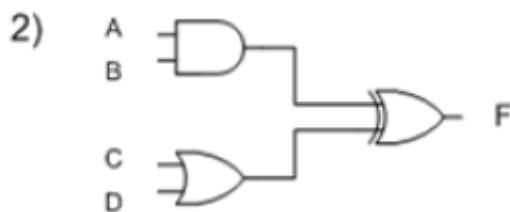
$$A \text{ NOR } B: \overline{[(\overline{A * A}) * (\overline{B * B})] * [(\overline{A * A}) * (\overline{B * B})]} = \overline{(\overline{A * A}) * (\overline{B * B})} = \overline{\overline{A} * \overline{A} * \overline{B} * \overline{B}} = \overline{\overline{A} * \overline{B}} = \overline{\overline{A + B}}.$$

$$(A \text{ NOR } B) \text{ AND } C = \overline{[(\overline{A + B}) * C] * [(\overline{A + B}) * C]} = \overline{(\overline{A + B}) * C} = \overline{\overline{A + B}} * C = A + B * C.$$

1)



(b)



$$A \text{ AND } B = A * B.$$

$$C \text{ OR } D = C + D.$$

$$(A \text{ AND } B) \text{ XOR } (C \text{ OR } D) = (A * B) (\overline{C + D}) + (\overline{A * B}) (C + D).$$

$$A \text{ AND } B = \overline{(\overline{A * B}) * (\overline{A * B})} = \overline{\overline{A} * \overline{B}} = A * B.$$

$$C \text{ OR } D = \overline{(\overline{C * C}) * (\overline{D * D})} = \overline{\overline{C} * \overline{D}} = \overline{\overline{C + D}} = C + D.$$

$$(A \text{ AND } B) \text{ XOR } (C \text{ OR } D) =$$

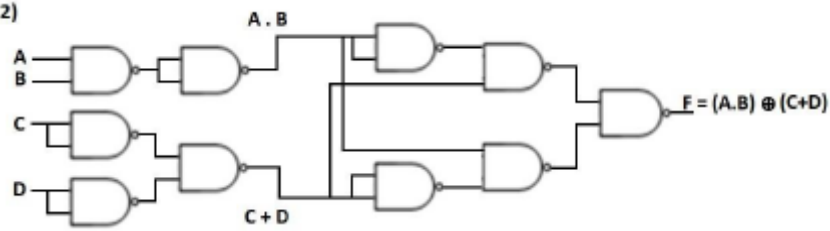
$$\overline{[(A * B) * (\overline{A * B}) * (C + D)] * [(A * B) * (\overline{C + D}) * (C + D)]} =$$

$$\overline{[(\overline{A * B}) * (C + D)] * [(\overline{A * B}) * (\overline{C + D})]} =$$

$$\overline{[(A * B) + (\overline{C + D})] * [(\overline{A * B}) + (C + D)]} =$$

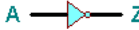



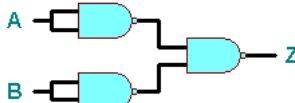





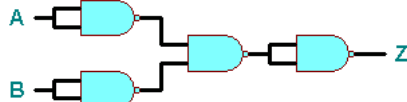

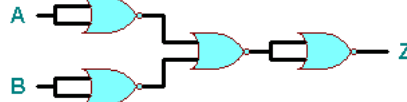

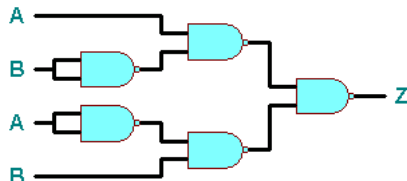
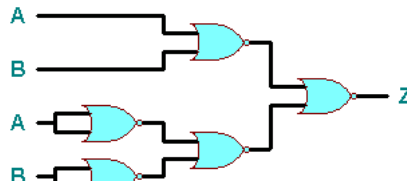

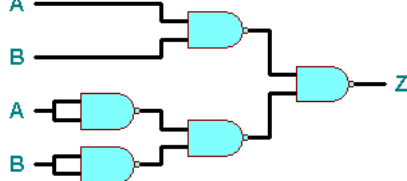
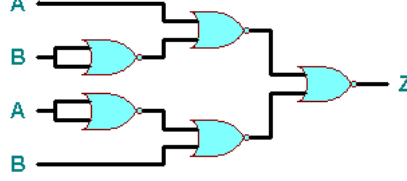
$$\overline{(A * B) + (\overline{C + D}) + (\overline{A * B}) + (C + D)} =$$
$$(A * B) (\overline{C + D}) + (\overline{A * B}) (C + D).$$

2)



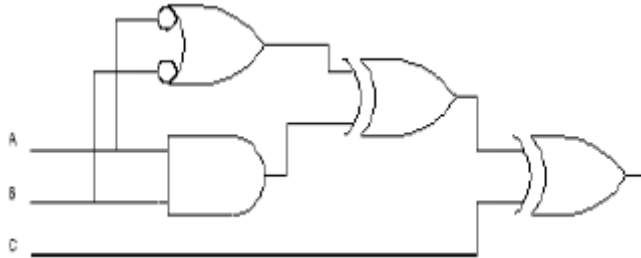
**Ejercicio 8.**

Reescribir las compuertas lógicas Not, Or, And y Xor utilizando, exclusivamente, compuertas NOR (ver como se resolvió el mismo caso para compuertas Nand, en Tener en ...).

| Compuerta |  | Equivalentes con compuertas NAND y NOR  |  |
|-----------|--|---|--|
|           |  | Nand  | Nor  |
| NOT       | <br>$Z = \bar{A}$   | <br>$Z = \overline{A \cdot A}$                                   | <br>$Z = \overline{A + A}$                          |
|           |  |   |  |
| OR        | <br>$Z = A + B$   | <br>$Z = \overline{\bar{A} \cdot \bar{B}}$                       | <br>$Z = \overline{\bar{A + B}}$                     |
|           |  |   |  |
| AND       | <br>$Z = A \cdot B$   | <br>$Z = \overline{\bar{A \cdot B}}$                             | <br>$Z = \overline{\bar{A + B}}$                     |
|           |  |   |  |
| NOR       | <br>$Z = \overline{A + B}$                                      | <br>$Z = \overline{\bar{A \cdot B}}$                           |  |
|           |  |   |  |
| NAND      | <br>$Z = \overline{A \cdot B}$                                  |   | <br>$Z = \overline{\bar{A + B}}$                   |
|           |  |   |  |
| XOR       | <br>$Z = A \oplus B$<br>$Z = A \cdot \bar{B} + \bar{A} \cdot B$ | <br>$Z = \overline{(A \cdot \bar{B}) \cdot (\bar{A} \cdot B)}$ | <br>$Z = \overline{(A + B) + (\bar{A} + \bar{B})}$ |
|           |  |   |  |
| XNOR      | <br>$Z = A \odot B$<br>$Z = A \cdot B + \bar{A} \cdot \bar{B}$  | <br>$Z = \overline{(A \cdot B) \cdot (\bar{A} \cdot \bar{B})}$ | <br>$Z = \overline{(A + \bar{B}) + (\bar{A} + B)}$ |
|           |  |   |  |

**Ejercicio 9.**

Construir la tabla de verdad del siguiente circuito. Analizar los valores y, basándose en sus conclusiones, construir un diagrama más simple que implemente la misma función de salida. Escribir, además, la ecuación de salida en forma de función.



| A | B | C | $\bar{A}$ | $\bar{B}$ | $\bar{A} \text{ OR } \bar{B}$ | A AND B | $(\bar{A} \text{ OR } \bar{B}) \text{ XOR } (A \text{ AND } B)$ | $[(\bar{A} \text{ OR } \bar{B}) \text{ XOR } (A \text{ AND } B)] \text{ XOR } C$ |
|---|---|---|-----------|-----------|-------------------------------|---------|---|--|
| 0 | 0 | 0 | 1         | 1         | 1                             | 0       | 1   | 1  |
| 0 | 0 | 1 | 1         | 1         | 1                             | 0       | 1   | 0  |
| 0 | 1 | 0 | 1         | 0         | 1                             | 0       | 1   | 1  |
| 0 | 1 | 1 | 1         | 0         | 1                             | 0       | 1   | 0  |
| 1 | 0 | 0 | 0         | 1         | 1                             | 0       | 1   | 1  |
| 1 | 0 | 1 | 0         | 1         | 1                             | 0       | 1   | 0  |
| 1 | 1 | 0 | 0         | 0         | 0                             | 1       | 1   | 1  |
| 1 | 1 | 1 | 0         | 0         | 0                             | 1       | 1   | 0  |

$$[(\bar{A} \text{ OR } \bar{B}) \text{ XOR } (A \text{ AND } B)] \text{ XOR } C = \text{NOT } C.$$

$$[(\bar{A} \text{ OR } \bar{B}) \text{ XOR } (A \text{ AND } B)] \text{ XOR } C = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C}.$$

$$[(\bar{A} \text{ OR } \bar{B}) \text{ XOR } (A \text{ AND } B)] \text{ XOR } C = (A + B + \bar{C}) (A + \bar{B} + \bar{C}) (\bar{A} + B + \bar{C}) (\bar{A} + \bar{B} + \bar{C}).$$

Gráfico.

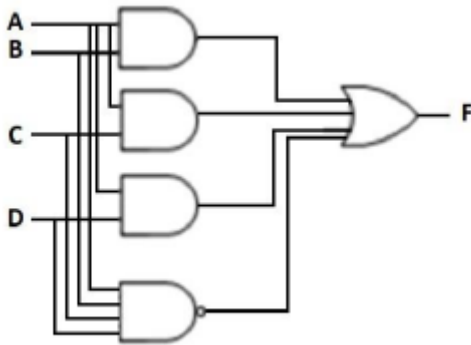


**Ejercicio 10.**

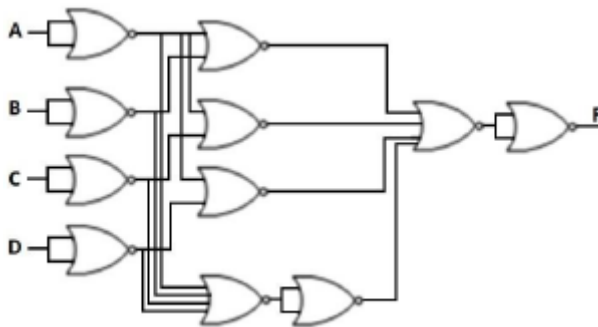
Dadas las siguientes relaciones, dibujar los diagramas de compuertas que cumplen con ellas. Modificarlos utilizando sólo compuertas NOR. Modificarlos utilizando sólo compuertas NAND.

(a)  $F = AB + AC + AD + \overline{ABCD}$ .

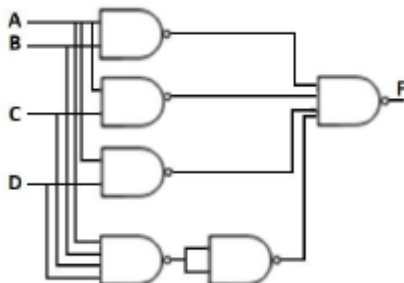
Original:



NOR:

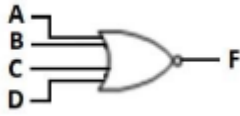


NAND:

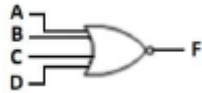


(b)  $F = \overline{A + B + C + D}$ .

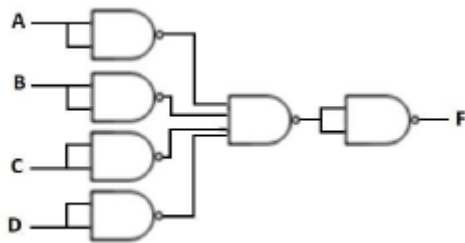
Original:



NOR:



NAND:



(c)  $F = \overline{A + B\bar{C}} + C$ .

Gráficos.

(d)  $F = A\bar{B} + \bar{A}B$ .

Gráficos.

**Ejercicio 11.**

Para la siguiente tabla de verdad, encontrar una fórmula lógica correspondiente (utilizando suma de productos).

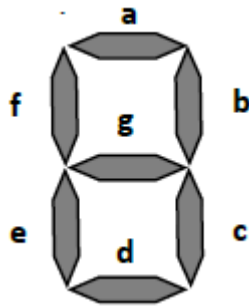
| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

$$F = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}C.$$

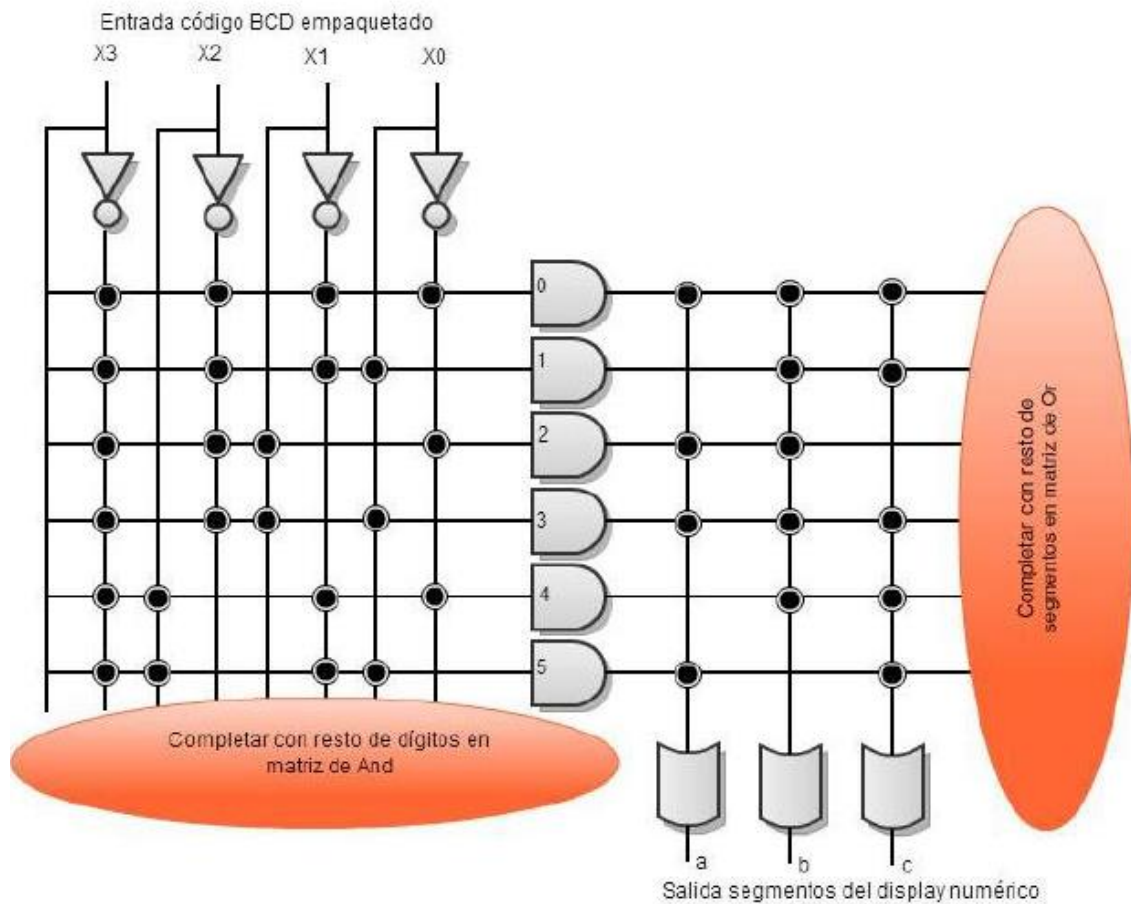
**Ejercicio 12.**

Diseñar un circuito que tenga como entrada código BCD empaquetado (4 entradas) y 7 salidas para controlar los 7 segmentos de un display numérico, siendo la salida para los segmentos “0” para apagado y “1” para prendido. Construir la tabla de verdad y la ecuación de la salida correspondiente a los segmentos a, b, c, d, e, f y g.

Ayuda 1: Cada segmento se considera como una salida distinta y cada uno se debe activar (poner en 1) dependiendo del número recibido en las entradas que representan los 4 bits de un BCD empaquetado. Ejemplo: El segmento b se debe activar cuando se recibe un 1 (0001), o un 2 (0010), o un 3 (0011), o un 4 (0100), o un 7 (0111), o un 8 (1000), o un 9 (1001). Se aplica la misma idea con el resto de las salidas.



Ayuda 2: Gráficamente, el circuito con las 4 entradas y las 7 salidas conviene diseñarlo como una matriz de compuertas And, seguida de la matriz de compuertas Or. Basarse en la siguiente gráfica parcial:



| $x_0$ | $x_1$ | $x_2$ | $x_3$ | a | b | c | d | e | f | g |
|-------|-------|-------|-------|---|---|---|---|---|---|---|
| 0     | 0     | 0     | 0     | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0     | 0     | 0     | 1     | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0     | 0     | 1     | 0     | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0     | 0     | 1     | 1     | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0     | 1     | 0     | 0     | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0     | 1     | 0     | 1     | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0     | 1     | 1     | 0     | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0     | 1     | 1     | 1     | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1     | 0     | 0     | 0     | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1     | 0     | 0     | 1     | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1     | 0     | 1     | 0     | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1     | 0     | 1     | 1     | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1     | 1     | 0     | 0     | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1     | 1     | 0     | 1     | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1     | 1     | 1     | 0     | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1     | 1     | 1     | 1     | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$$a = \overline{x_0}x_1x_2x_3 + x_0\overline{x_1}x_2x_3 + \overline{x_0}\overline{x_1}x_2x_3 + \overline{x_0}x_1\overline{x_2}x_3 + \overline{x_0}x_1x_2\overline{x_3} + \overline{x_0}x_1x_2x_3 + x_0\overline{x_1}\overline{x_2}x_3 + x_0\overline{x_1}x_2x_3.$$

$$b = \overline{x_0}x_1x_2x_3 + \overline{x_0}\overline{x_1}\overline{x_2}x_3 + \overline{x_0}\overline{x_1}x_2\overline{x_3} + \overline{x_0}x_1\overline{x_2}x_3 + \overline{x_0}x_1x_2\overline{x_3} + \overline{x_0}x_1x_2x_3 + x_0\overline{x_1}\overline{x_2}x_3 + x_0\overline{x_1}x_2x_3.$$

$$c = \overline{x_0 x_1 x_2 x_3} + \overline{x_0 x_1 x_2} x_3 + \overline{x_0 x_1} x_2 x_3 + \overline{x_0 x_1} \overline{x_2 x_3} + \overline{x_0 x_1} \overline{x_2} x_3 + \overline{x_0 x_1} x_2 \overline{x_3} + \overline{x_0 x_1} x_2 x_3 + x_0 \overline{x_1 x_2 x_3} + x_0 \overline{x_1} x_2 x_3.$$

$$d = \overline{x_0 x_1 x_2 x_3} + \overline{x_0 x_1} x_2 \overline{x_3} + \overline{x_0 x_1} x_2 x_3 + \overline{x_0 x_1} \overline{x_2} x_3 + \overline{x_0 x_1} x_2 \overline{x_3} + x_0 \overline{x_1} x_2 \overline{x_3}.$$

$$e = \overline{x_0 x_1 x_2 x_3} + \overline{x_0 x_1} x_2 \overline{x_3} + \overline{x_0 x_1} x_2 x_3 + x_0 \overline{x_1} x_2 \overline{x_3}.$$

$$f = \overline{x_0 x_1} x_2 \overline{x_3} + \overline{x_0 x_1} x_2 x_3 + \overline{x_0 x_1} \overline{x_2} x_3 + \overline{x_0 x_1} x_2 \overline{x_3} + x_0 \overline{x_1} x_2 \overline{x_3} + x_0 \overline{x_1} x_2 x_3.$$

$$g = \overline{x_0 x_1} x_2 \overline{x_3} + \overline{x_0 x_1} x_2 x_3 + \overline{x_0 x_1} \overline{x_2} x_3 + \overline{x_0 x_1} x_2 \overline{x_3} + \overline{x_0 x_1} \overline{x_2} x_3 + \overline{x_0 x_1} x_2 \overline{x_3} + x_0 \overline{x_1} x_2 \overline{x_3} + x_0 \overline{x_1} \overline{x_2} x_3.$$

**Ejercicio 13.**

Un controlador de proceso industrial recibe como entrada tres señales de temperatura  $T1$ ,  $T2$ ,  $T3$  ( $T1 < T2 < T3$ ) que adoptan el valor lógico “1” cuando la temperatura es mayor que  $t1$ ,  $t2$  y  $t3$ , respectivamente. Diseñar un circuito que genere una señal  $F$  cuando la temperatura esté comprendida entre  $t1$  y  $t2$  o cuando la temperatura sea mayor que  $t3$ .

| T1 | T2 | T3 | F |
|----|----|----|---|
| 0  | 0  | 0  | 0 |
| 0  | 0  | 1  | 1 |
| 0  | 1  | 0  | 0 |
| 0  | 1  | 1  | 1 |
| 1  | 0  | 0  | 1 |
| 1  | 0  | 1  | 1 |
| 1  | 1  | 0  | 0 |
| 1  | 1  | 1  | 1 |

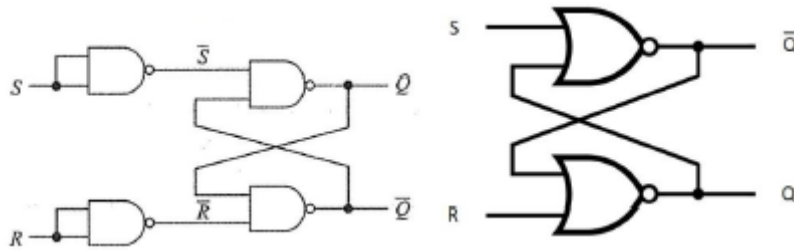
$$F = T1\overline{T2}\overline{T3} + T1T2T3.$$

$$F = (T1 + T2 + T3)(\overline{T1} + \overline{T2} + T3).$$

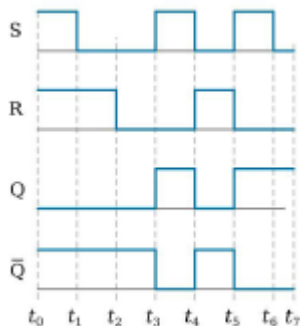
**Ejercicio 14.**

Dibujar el esquema de compuertas que componen un flip-flop S-R. Describir, a través de una tabla, los estados en función de las entradas. Modificar el esquema anterior para hacerlo síncrono. Describir, gráficamente, su respuesta temporal.

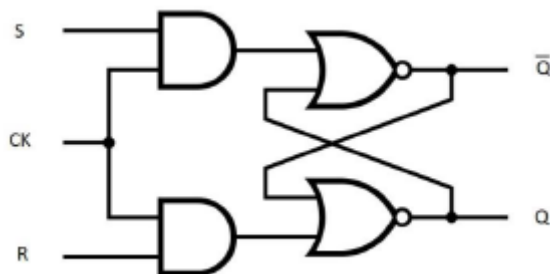
Asincrónico:



| S | R | $Q_{t+1}$ |
|---|---|-----------|
| 0 | 0 | $Q_t$     |
| 0 | 1 | 0         |
| 1 | 0 | 1         |
| 1 | 1 | Prohibido |

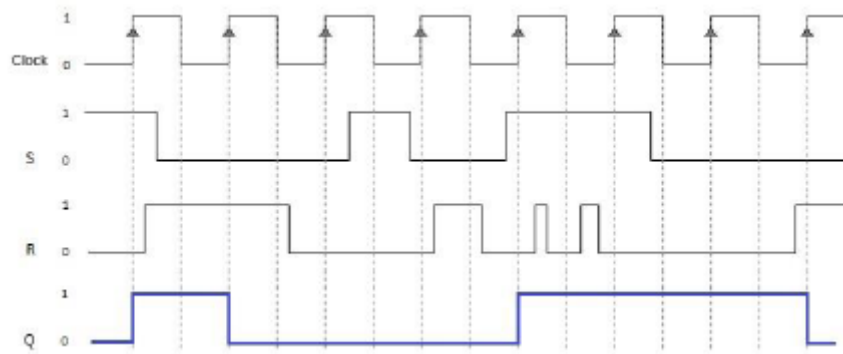


Síncrono:



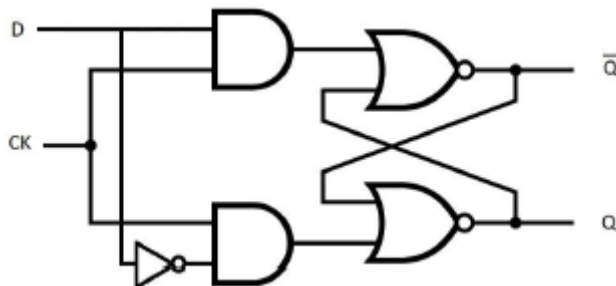


| S | R | CLK | $Q_{t+1}$ |
|---|---|-----|-----------|
| 0 | 0 | 1   | $Q_t$     |
| 0 | 1 | 1   | 0         |
| 1 | 0 | 1   | 1         |
| 1 | 1 | 1   | Prohibido |
| x | x | 0   | $Q_t$     |

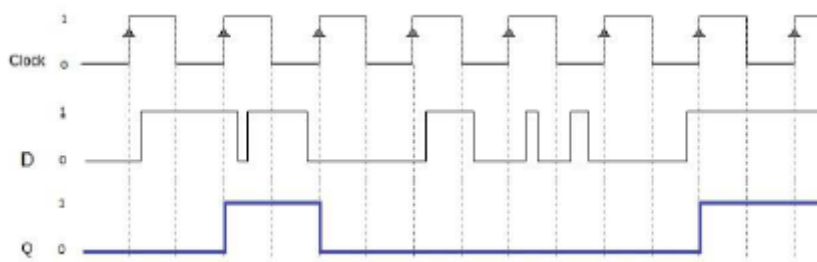


**Ejercicio 15.**

Dibujar el esquema de un flip-flop D. Detallar, en su respuesta temporal, cómo resuelve el problema de la doble entrada de 1's que se presentaba en el S-R.

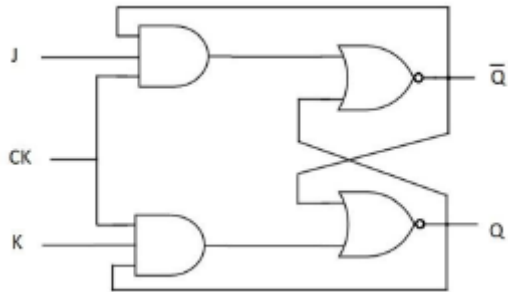


| D | CLK | $Q_{t+1}$ |
|---|-----|-----------|
| 0 | 1   | 0         |
| 1 | 1   | 1         |
| x | 0   | $Q_t$     |

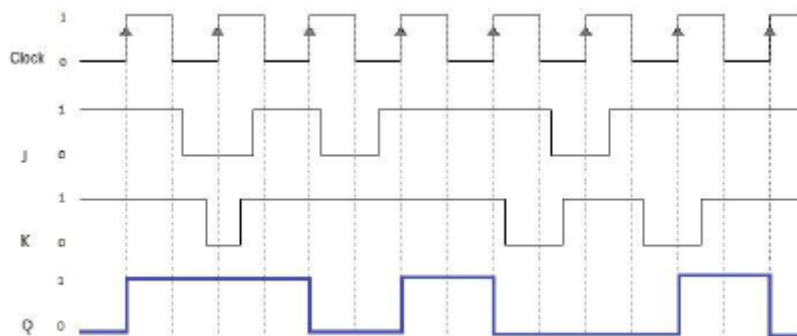


### Ejercicio 16.

Dibujar el esquema de un flip-flop J-K, describiendo su respuesta temporal.

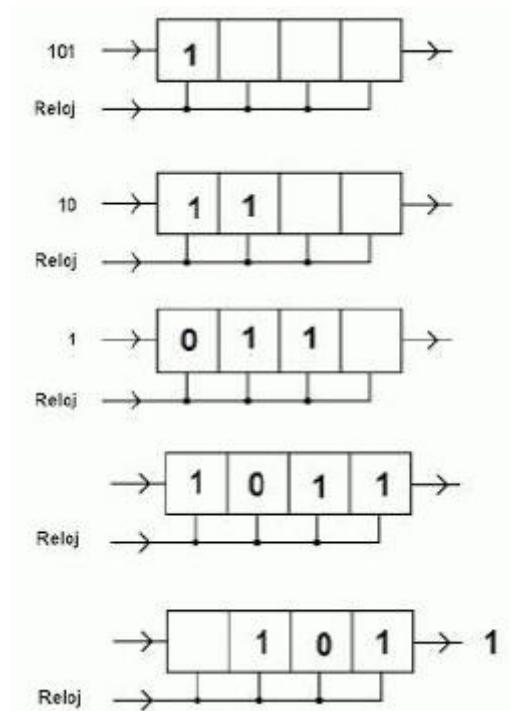


| J | K | CLK | $Q_{t+1}$   |
|---|---|-----|-------------|
| 0 | 0 | 1   | $Q_t$       |
| 0 | 1 | 1   | 0           |
| 1 | 0 | 1   | 1           |
| 1 | 1 | 1   | $\bar{Q}_t$ |
| x | x | 0   | $Q_t$       |



**Ejercicio 17.**

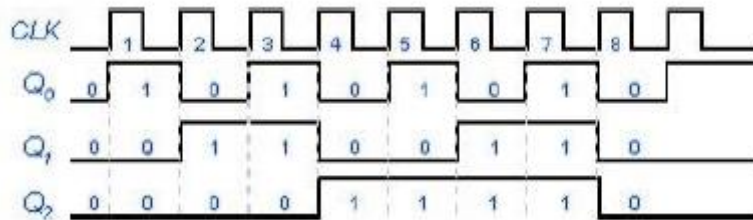
Dibujar el diagrama de tiempos del registro de la figura, implementado con flip-flops D. Modificarlo para desplazamiento izquierda derecha y derecha izquierda. Ayuda: Ejemplo de respuesta temporal para interpretar cómo responde el registro previo ante la entrada serial del número binario 1011:



Gráficos.

**Ejercicio 18.**

Describir, gráficamente, la respuesta temporal de cada flip-flop ante una señal de unos y ceros entrando por reloj. Ayuda: El diagrama correspondiente considerando sólo los primeros 3 flip-flops es el siguiente:



Se observa cómo la respuesta de cada flip-flop emite una onda a la mitad de frecuencia que su clock de entrada.

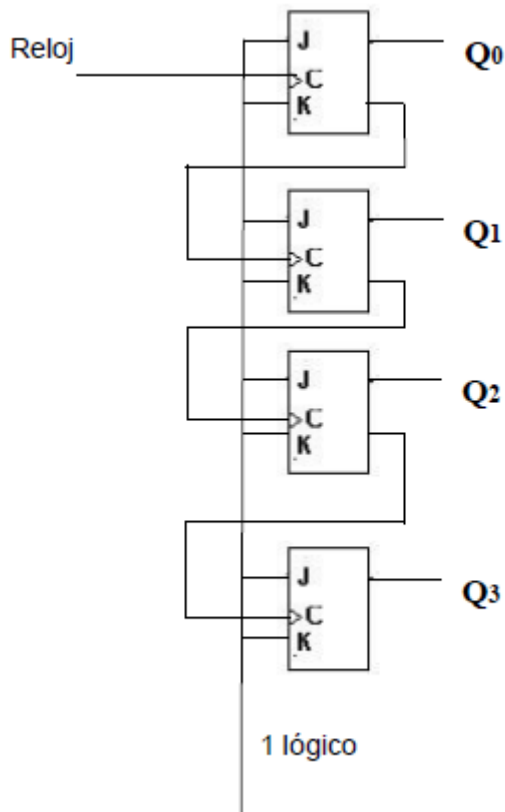


Gráfico.

Objetivos de la práctica: que el alumno domine

- Las instrucciones básicas del lenguaje assembly del simulador MSX88.
- Los diferentes modos de direccionamiento.
- El diseño de programas utilizando instrucciones de salto condicional.
- Realice el diseño de programas utilizando instrucciones del MSX88.
- Comprenda la utilidad y funcionamiento de las subrutinas.

Bibliografía:

- Apunte 4 de la cátedra, "Lenguaje Assembly".
- Manual del simulador MSX88.
- Set de Instrucciones de MSX88.

1) Dada la siguiente definición de datos y el código:  $F = [(A+B)/C] - D$

| nombre | tamaño | valor |
|--------|--------|-------|
| A:     | 1 byte | 6     |
| B:     | 1 byte | 4     |
| C:     | 1 byte | 2     |
| D:     | 1 byte | 1     |
| F:     | 1 byte | ?     |

Suponiendo que se poseen las instrucciones necesarias en cada caso, escribir el programa que implemente el código anterior utilizando máquinas de 1, 2 ó 3 direcciones.

| Maq. de 1 dirección | Maq. de 2 direcciones | Maq. de 3 direcciones |
|---------------------|-----------------------|-----------------------|
|                     |                       |                       |

2) Suponga que cada código de operación ocupa 6 bits y las direcciones son de 10 bits. Analice las soluciones implementadas en el ejercicio anterior y complete la siguiente tabla:

|  | M. de 1 dirección | M. de 2 direcciones | M. de 3 direcciones |
|--|-------------------|---------------------|---------------------|
| Tamaño del programa en memoria (cod.operación + operandos) |                   |                     |                     |
| Cantidad de accesos a memoria (instrucciones + operandos)  |                   |                     |                     |

3) Dado el siguiente código:  $F = ((A - B) * C) + (D/E)$ ;

- Implemente el código utilizando máquinas de 1, 2 y 3 direcciones.
- Realice una tabla de comparación similar a la del ejercicio 2.
- ¿Cuál máquina elegiría haciendo un balance de la cantidad de instrucciones, el espacio en memoria ocupado y el tiempo de ejecución (1 acceso a memoria = 1 ms)? ¿Es ésta una conclusión general?

Para cada programa propuesto en los siguientes ejercicios, deberá editar el archivo fuente con extensión asm (ej: ejer1.asm), luego ensamblarlo usando asm88.exe (comando: asm88 ejer1.asm) y enlazarlo con link88.exe (comando: link88 ejer1.o). Cada archivo obtenido con extensión eje (ej: ejer1.eje) deberá ser cargado y ejecutado en el simulador MSX88.

4) El siguiente programa utiliza una **instrucción de transferencia de datos** (instrucción MOV) con diferentes modos de direccionamiento para referenciar sus operandos. Ejecutar y analizar el funcionamiento de cada instrucción en el Simulador MSX88 observando el flujo de información a través del BUS DE DATOS, el BUS DE DIRECCIONES, el BUS DE CONTROL, el contenido de REGISTROS, de posiciones de MEMORIA, operaciones en la ALU, etc.

|                |                           |
|----------------|---------------------------|
| ORG 1000h      | ORG 2000h                 |
| NUM0 DB 0CAh   | MOV BL, NUM0              |
| NUM1 DB 0      | MOV BH, 0FFh              |
| NUM2 DW ?      | MOV CH, BL                |
| NUM3 DW 0ABCDh | MOV AX, BX                |
| NUM4 DW ?      | MOV NUM1, AL              |
|                | MOV NUM2, 1234h           |
|                | MOV BX, OFFSET NUM3       |
|                | MOV DL, [BX]              |
|                | MOV AX, [BX]              |
|                | MOV BX, 1006h             |
|                | MOV WORD PTR [BX], 0CDEFh |
|                | HLT                       |

# Organización de Computadoras 2023

END

## Cuestionario:

- Explicar detalladamente qué hace cada instrucción MOV del programa anterior, en función de sus operandos y su modo de direccionamiento.
  - Confeccionar una tabla que contenga todas las instrucciones MOV anteriores, el modo de direccionamiento y el contenido final del operando destino de cada una de ellas.
  - Notar que durante la ejecución de algunas instrucciones MOV aparece en la pantalla del simulador un registro temporal denominado “ri”, en ocasiones acompañado por otro registro temporal denominado “id”. Explicar con detalle que función cumplen estos registros.
- 5) El siguiente programa utiliza diferentes **instrucciones de procesamiento de datos** (instrucciones aritméticas y lógicas). Analice el comportamiento de ellas y ejecute el programa en el MSX88.

|                    |                   |
|--------------------|-------------------|
| ORG 1000H          | ORG 2000H         |
| NUM0 DB 80h        | MOV AL, NUM0      |
| NUM1 DB 200        | ADD AL, AL        |
| NUM2 DB -1         | INC NUM1          |
| BYTE0 DB 01111111B | MOV BH, NUM1      |
| BYTE1 DB 10101010B | MOV BL, BH        |
|                    | DEC BL            |
|                    | SUB BL, BH        |
|                    | MOV CH, BYTE1     |
|                    | AND CH, BYTE0     |
|                    | NOT BYTE0         |
|                    | OR CH, BYTE0      |
|                    | XOR CH, 11111111B |
|                    | HLT               |
|                    | END               |

## Cuestionario:

- ¿Cuál es el estado de los FLAGS después de la ejecución de las instrucciones ADD y SUB del programa anterior? Justificar el estado (1 ó 0) de cada uno de ellos. ¿Dan alguna indicación acerca de la correctitud de los resultados?
  - ¿Qué cadenas binarias representan a NUM1 y NUM2 en la memoria del simulador? ¿En qué sistemas binarios están expresados estos valores?
  - Confeccionar una tabla que indique para cada operación aritmética ó lógica del programa, el valor de sus operandos, en qué registro o dirección de memoria se almacenan y el resultado de cada operación.
- 6) El siguiente programa implementa un contador utilizando una **instrucción de transferencia de control**. Analice el funcionamiento de cada instrucción y en particular las del lazo repetitivo que provoca la cuenta.

|           |              |
|-----------|--------------|
| ORG 1000H | ORG 2000H    |
| INI DB 0  | MOV AL, INI  |
| FIN DB 15 | MOV AH, FIN  |
|           | SUMA: INC AL |
|           | CMP AL, AH   |
|           | JNZ SUM      |
|           | HLT          |
|           | END          |

## Cuestionario:

- ¿Cuántas veces se ejecuta el lazo? ¿De qué variables depende esto en el caso general?
  - Analice y ejecute el programa reemplazando la instrucción de salto condicional JNZ por las siguientes, indicando en cada caso el contenido final del registro AL:
    - JS
    - JZ
    - JMP
- 7) Escribir un programa en lenguaje assembly del MSX88 que implemente la sentencia condicional de un lenguaje de alto nivel IF **A < B** THEN **C = A** ELSE **C = B**. Considerar que las variables de la sentencia están almacenadas en los registros internos de la CPU del siguiente modo **A** en AL, **B** en BL y **C** en CL.  
Determine las modificaciones que debería hacer al programa si la condición de la sentencia IF fuera:

# Organización de Computadoras 2023

a)  $A \leq B$

b)  $A = B$

- 8) El siguiente programa suma todos los elementos de una tabla almacenada a partir de la dirección 1000H de la memoria del simulador. Analice el funcionamiento y determine el resultado de la suma. Comprobar resultado en el MSX88.

```
ORG 1000H
TABLA DB 2,4,6,8,10,12,14,16,18,20
FIN DB ?
TOTAL DB ?
MAX DB 13
```

```
ORG 2000H
MOV AL, 0
MOV CL, OFFSET FIN - OFFSET TABLA
MOV BX, OFFSET TABLA
SUMA: ADD AL, [BX]
      INC BX
      DEC CL
      JNZ SUMA
      HLT
      END
```

¿Qué modificaciones deberá hacer en el programa para que el mismo almacene el resultado de la suma en la celda etiquetada TOTAL?

- 9) Escribir un programa que, utilizando las mismas variables y datos que el programa del punto anterior (TABLA, FIN, TOTAL, MAX), determine cuántos de los elementos de TABLA son menores o iguales que MAX. Dicha cantidad debe almacenarse en la celda TOTAL.

- 10) Analizar el funcionamiento del siguiente programa.

```
ORG 2000H
MOV AX, 1
MOV BX, 1000h
CARGA: MOV [BX], AX
      ADD BX, 2
      ADD AX, AX
      CMP AX, 200
      JS CARGA
      HLT
      END
```

Cuestionario:

- El programa genera una tabla. ¿Cómo están relacionados sus elementos entre sí?
  - ¿A partir de qué dirección de memoria se crea la tabla? ¿Cuál es la longitud de cada uno de sus elementos (medida en bits)?
  - ¿Cuántos elementos tiene la tabla una vez finalizada la ejecución del programa? ¿De qué depende esta cantidad?
- 11) Escribir un programa que genere una tabla a partir de la dirección de memoria almacenada en la celda DIR con los múltiplos de 5 desde cero hasta MAX.
- 12) Escribir un programa que, dado un número X, genere un arreglo con todos los resultados que se obtienen hasta llegar a 0, aplicando la siguiente fórmula: si X es par, se le resta 7; si es impar, se le suma 5, y al resultado se le aplica nuevamente la misma fórmula. Ej: si  $X = 3$  entonces el arreglo tendrá: 8, 1, 6, -1, 4, -3, 2, -5, 0.
- 13) Dada la frase "Organización y la Computación", almacenada en la memoria, escriba un programa que determine cuantas letras 'a' seguidas de 'c' hay en ella.
- 14) Escribir un programa que sume dos números representados en Ca2 de 32 bits almacenados en memoria de datos y etiquetados NUM1 y NUM2 y guarde el resultado en RESUL (en este caso cada dato y el resultado ocuparán 4 celdas consecutivas de memoria). Verifique el resultado final y almacene 0FFH en la celda BIEN en caso de ser correcto o en otra MAL en caso de no serlo. Recordar que el MSX88 trabaja con números en Ca2 pero tener en cuenta que las operaciones con los 16 bits menos significativos de cada número deben realizarse en BSS.
- 15) Escribir un programa que efectúe la suma de dos vectores de 6 elementos cada uno (donde cada elemento es un número de 32 bits) almacenados en memoria de datos y etiquetados TAB1 y TAB2 y guarde el resultado en TAB3. Suponer en primera instancia que no existirán errores de tipo aritmético (ni carry ni overflow), luego analizar y definir los cambios y agregados necesarios que deberían realizarse al programa para tenerlos en cuenta.



## Organización de Computadoras 2023

16) Los siguientes programas realizan la misma tarea, en uno de ellos se utiliza una **instrucción de transferencia de control con retorno**. Analícelos y compruebe la equivalencia funcional.

```
        ; Memoria de Datos
        ORG 1000H
NUM1 DB 5H
NUM2 DB 3H

        ; Memoria de Instrucciones
        ORG 2000H
        MOV AL, NUM1
        CMP AL, 0
        JZ FIN
        MOV AH, 0
        MOV DX, 0
        MOV CL, NUM2
LOOP:   CMP CL, 0
        JZ FIN
        ADD DX, AX
        DEC CL
        JMP LOOP
FIN:    HLT
        END
```

```
        ; Memoria de Datos
        ORG 1000H
NUM1 DB 5H
NUM2 DB 3H

        ; Memoria de Instrucciones
        ORG 3000H ; Subrutina SUB1
SUB1:   CMP AL, 0
        JZ FIN
        CMP CL, 0
        JZ FIN
        MOV AH, 0
        MOV DX, 0
LAZO:   ADD DX, AX
        DEC CX
        JNZ LAZO
FIN:    RET

        ORG 2000H ; Programa principal
        MOV AL, NUM1
        MOV CL, NUM2
        CALL SUB1
        HLT
        END
```

Responder:

- 1) ¿Cuál es la tarea realizada por ambos programas?
- 2) ¿Dónde queda almacenado el resultado?
- 3) ¿Cuál programa realiza la tarea más rápido? ¿El tiempo de ejecución de la tarea depende de los valores almacenados en NUM1, en NUM2, en ambos lugares o en ninguno?

Explicar detalladamente:

- a) Todas las acciones que tienen lugar al ejecutarse la instrucción CALL SUB1.
- b) ¿Qué operación se realiza con la instrucción RET?, ¿cómo sabe la CPU a qué dirección de memoria debe retornar desde la subrutina al programa principal?

El siguiente programa es otra forma de implementación de la tarea del punto anterior (ejercicio 16). Analizar y establecer las diferencias con las anteriores, en particular las relacionadas a la forma de 'proveer' los operandos a las subrutinas.

```
        ; Memoria de datos
        ORG 1000H
NUM1 DW 5H ; NUM1 y NUM2 deben ser mayores que cero
NUM2 DW 3H

        ; Memoria de Instrucciones
        ORG 3000H ; Subrutina SUB2
SUB2:   MOV DX, 0
LAZO:   MOV BX, AX
        ADD DX, [BX]
        PUSH DX
        MOV BX, CX
        MOV DX, [BX]
        DEC DX
        MOV [BX], DX
        POP DX
        JNZ LAZO
        RET

        ORG 2000H ; Programa principal
        MOV AX, OFFSET NUM1
        MOV CX, OFFSET NUM2
        CALL SUB2
        HLT
        END
```

# Organización de Computadoras 2023

Explicar detalladamente:

- a) Todas las acciones que tienen lugar al ejecutarse las instrucciones PUSH DX y POP DX.
- b) Cuáles son los dos usos que tiene el registro DX en la subrutina SUB2.

- 18) Escribir un programa que sume 2 vectores de 6 elementos (similar al realizado en el ejercicio 15), de modo tal que utilice una subrutina que sume números de 32 bits (similar al programa escrito en ejercicio 14).
- 19) Escriba una subrutina que reciba la mantisa entera en BSS y el exponente en BSS de un número en los registros AH y AL respectivamente y devuelva, en ellos, una representación equivalente del mismo pero con el exponente disminuido en 1 y la mantisa ajustada. De no ser posible el ajuste, BL debe contener 0FFH en vez de 00H en el retorno.
- 20) Escriba una subrutina que reciba como parámetro un número en el formato IEEE 754 de simple precisión y analice/verifique las características del mismo devolviendo en el registro CL un valor igual a 0 si el número está sin normalizar, 1 en caso de ser +/- infinito, 2 si es un NAN, 3 si es un +/- cero y 4 si es un número normalizado. La subrutina recibe en AX la parte alta del número y en BX la parte baja.
- 21) Modifique la subrutina del ejercicio 19 para el caso en que la mantisa y el exponente estén representados en BCS.

## Datos útiles:

- Las subrutinas siempre se escriben antes que el programa principal, aunque su dirección de comienzo sea más alta.
- Las etiquetas de subrutinas y bucles van seguidas de dos puntos (:).
- Los operandos en hexadecimal terminan en H y los que comienzan con una letra van precedidos por un cero (0) para no ser confundidos con etiquetas (por ejemplo, 0A4H en lugar de A4H).
- Se pueden incluir comentarios en los programas, anteponiendo siempre un punto y coma (;).
- El direccionamiento indirecto solo está implementado con el registro BX.
- Cada celda de memoria almacena un byte. Los datos de dos bytes (words) se almacenan de la siguiente manera: primero la parte baja (byte menos significativo) y luego la parte alta. Esto se corresponde con la idea de que la parte baja del dato se almacena en la dirección más baja y la parte alta, en la dirección más alta.

## **Trabajo Práctico N° 4:** **Assembly, Instrucciones y Simulador MSX88.**

### **Ejercicio 1.**

Dada la siguiente definición de datos y el código  $F = \frac{A+B}{C} - D$ .

| Nombre | Tamaño | Valor |
|--------|--------|-------|
| A      | 1 byte | 6     |
| B      | 1 byte | 4     |
| C      | 1 byte | 2     |
| D      | 1 byte | 1     |
| F      | 1 byte | ?     |

Suponiendo que se poseen las instrucciones necesarias en cada caso, escribir el programa que implemente el código anterior utilizando máquinas de 1, 2 o 3 direcciones.

#### **Máquina de 1 dirección:**

load A  
add B  
div C  
sub D  
store F

Instrucciones: 5.

Tamaño del programa en memoria: 10 bytes.

Accesos: 10 MI y 5 MD.

#### **Máquina de 2 dirección:**

mov F, A  
add F, B  
div F, C  
sub F, D

Instrucciones: 4.

Tamaño del programa en memoria: 12 bytes.

Accesos: 12 MI y 11 MD.

#### **Máquina de 3 dirección:**

add F, A, B  
div F, F, C  
sub F, F, D

Instrucciones: 3.

Tamaño del programa en memoria: 12 bytes.

Accesos: 12 MI y 9 MD.

**Ejercicio 2.**

Suponer que cada código de operación ocupa 6 bits y las direcciones son de 10 bits. Analizar las soluciones implementadas en el ejercicio anterior y completar la siguiente tabla:

|  | Máquina de 1<br>dirección  | Máquina de 2<br>direcciones | Máquina de 3<br>direcciones |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
|--|--|-----------------------------|-----------------------------|-------|----------|-------|----------|-------|----------|---------|----------|-------|----------|---|----------|----------|----------|----------|----------|----------|----------|----------|-------|----------|--|----------------|-----------|----------------|-----------|----------------|-----------|-------|----------|
| Tamaño del<br>programa en<br>memoria (código<br>de operación más<br>operandos) | <table><tr><td>load A</td><td>1 + 2= 3</td></tr><tr><td>add B</td><td>1 + 2= 3</td></tr><tr><td>div C</td><td>1 + 2= 3</td></tr><tr><td>sub D</td><td>1 + 2= 3</td></tr><tr><td>store F</td><td>1 + 2= 3</td></tr><tr><td>total</td><td>15 bytes</td></tr></table> | load A                      | 1 + 2= 3                    | add B | 1 + 2= 3 | div C | 1 + 2= 3 | sub D | 1 + 2= 3 | store F | 1 + 2= 3 | total | 15 bytes | <table><tr><td>mov F, A</td><td>1 + 4= 5</td></tr><tr><td>add F, B</td><td>1 + 4= 5</td></tr><tr><td>div F, C</td><td>1 + 4= 5</td></tr><tr><td>sub F, D</td><td>1 + 4= 5</td></tr><tr><td>total</td><td>20 bytes</td></tr></table> | mov F, A | 1 + 4= 5 | add F, B | 1 + 4= 5 | div F, C | 1 + 4= 5 | sub F, D | 1 + 4= 5 | total | 20 bytes | <table><tr><td>add F, A,<br/>B</td><td>1 + 6= 7</td></tr><tr><td>div F, F,<br/>C</td><td>1 + 6= 7</td></tr><tr><td>sub F, F,<br/>D</td><td>1 + 6= 7</td></tr><tr><td>total</td><td>21 bytes</td></tr></table>    | add F, A,<br>B | 1 + 6= 7  | div F, F,<br>C | 1 + 6= 7  | sub F, F,<br>D | 1 + 6= 7  | total | 21 bytes |
| load A   | 1 + 2= 3   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| add B  | 1 + 2= 3   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| div C  | 1 + 2= 3   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| sub D  | 1 + 2= 3   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| store F  | 1 + 2= 3   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| total  | 15 bytes   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| mov F, A   | 1 + 4= 5   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| add F, B   | 1 + 4= 5   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| div F, C   | 1 + 4= 5   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| sub F, D   | 1 + 4= 5   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| total  | 20 bytes   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| add F, A,<br>B   | 1 + 6= 7   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| div F, F,<br>C   | 1 + 6= 7   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| sub F, F,<br>D   | 1 + 6= 7   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| total  | 21 bytes   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| Cantidad de<br>accesos a memoria<br>(instrucciones +<br>operandos)             | <table><tr><td>load A</td><td>3 + 1= 4</td></tr><tr><td>add B</td><td>3 + 1= 4</td></tr><tr><td>div C</td><td>3 + 1= 4</td></tr><tr><td>sub D</td><td>3 + 1= 4</td></tr><tr><td>store F</td><td>3 + 1= 4</td></tr><tr><td>total</td><td>20 bytes</td></tr></table> | load A                      | 3 + 1= 4                    | add B | 3 + 1= 4 | div C | 3 + 1= 4 | sub D | 3 + 1= 4 | store F | 3 + 1= 4 | total | 20 bytes | <table><tr><td>mov F, A</td><td>5 + 2= 7</td></tr><tr><td>add F, B</td><td>5 + 3= 8</td></tr><tr><td>div F, C</td><td>5 + 3= 8</td></tr><tr><td>sub F, D</td><td>5 + 3= 8</td></tr><tr><td>total</td><td>31 bytes</td></tr></table> | mov F, A | 5 + 2= 7 | add F, B | 5 + 3= 8 | div F, C | 5 + 3= 8 | sub F, D | 5 + 3= 8 | total | 31 bytes | <table><tr><td>add F, A,<br/>B</td><td>7 + 3= 10</td></tr><tr><td>div F, F,<br/>C</td><td>7 + 3= 10</td></tr><tr><td>sub F, F,<br/>D</td><td>7 + 3= 10</td></tr><tr><td>total</td><td>30 bytes</td></tr></table> | add F, A,<br>B | 7 + 3= 10 | div F, F,<br>C | 7 + 3= 10 | sub F, F,<br>D | 7 + 3= 10 | total | 30 bytes |
| load A   | 3 + 1= 4   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| add B  | 3 + 1= 4   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| div C  | 3 + 1= 4   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| sub D  | 3 + 1= 4   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| store F  | 3 + 1= 4   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| total  | 20 bytes   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| mov F, A   | 5 + 2= 7   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| add F, B   | 5 + 3= 8   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| div F, C   | 5 + 3= 8   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| sub F, D   | 5 + 3= 8   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| total  | 31 bytes   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| add F, A,<br>B   | 7 + 3= 10  |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| div F, F,<br>C   | 7 + 3= 10  |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| sub F, F,<br>D   | 7 + 3= 10  |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |
| total  | 30 bytes   |                             |                             |       |          |       |          |       |          |         |          |       |          |   |          |          |          |          |          |          |          |          |       |          |  |                |           |                |           |                |           |       |          |

**Ejercicio 3.**

Dado el siguiente código:  $F = (A - B) C + \frac{D}{E}$ .

(a) Implementar el código utilizando máquinas de 1, 2 y 3 direcciones.

**Máquina de 1 dirección:**

```
load A
sub B
mul C
store A
load D
div E
add A
store F
```

Instrucciones: 8.

Tamaño del programa en memoria: 16 bytes.

Accesos: 16 MI y 8 MD.

**Máquina de 2 dirección:**

```
mov F, A
sub F, B
mul F, C
div D, E
add F, D
```

Instrucciones: 5.

Tamaño del programa en memoria: 15 bytes.

Accesos: 15 MI y 14 MD.

**Máquina de 3 dirección:**

```
sub F, A, B
mul F, F, C
div D, D, E
add F, F, D
```

Instrucciones: 4.

Tamaño del programa en memoria: 16 bytes.

Accesos: 16 MI y 12 MD.

(b) Realizar una tabla de comparación similar a la del Ejercicio 2.

|  | Máquina de 1<br>dirección |                 | Máquina de 2<br>direcciones  |          | Máquina de 3<br>direcciones |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|--|---------------------------|-----------------|--|----------|-----------------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|--------------|-----------------|----------------|----------|
| Tamaño del<br>programa en<br>memoria (código<br>de operación más<br>operandos) | load A                    | 1 + 1= 2        | <table><tr><td>mov F, A</td><td>1 + 2= 3</td></tr><tr><td>sub F, B</td><td>1 + 2= 3</td></tr><tr><td>mul F, C</td><td>1 + 2= 3</td></tr><tr><td>div D, E</td><td>1 + 2= 3</td></tr><tr><td>add F, D</td><td>1 + 2= 3</td></tr><tr><td><b>total</b></td><td><b>15 bytes</b></td></tr></table> |          | mov F, A                    | 1 + 2= 3 | sub F, B | 1 + 2= 3 | mul F, C | 1 + 2= 3 | div D, E | 1 + 2= 3 | add F, D | 1 + 2= 3 | <b>total</b> | <b>15 bytes</b> | sub F, A,<br>B | 1 + 3= 4 |
|  | mov F, A                  | 1 + 2= 3        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | sub F, B                  | 1 + 2= 3        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | mul F, C                  | 1 + 2= 3        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | div D, E                  | 1 + 2= 3        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | add F, D                  | 1 + 2= 3        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | <b>total</b>              | <b>15 bytes</b> |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | sub B                     | 1 + 1= 2        | mul F, F,<br>C   | 1 + 3= 4 |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| mul C  | 1 + 1= 2                  | div D, D,<br>E  | 1 + 3= 4   |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| store A  | 1 + 1= 2                  | add F, F,<br>D  | 1 + 3= 4   |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| load D   | 1 + 1= 2                  | <b>total</b>    | <b>16 bytes</b>  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| div E  | 1 + 1= 2                  |                 |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| add A  | 1 + 1= 2                  |                 |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| store F  | 1 + 1= 2                  |                 |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| <b>total</b>   | <b>16 bytes</b>           |                 |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| Cantidad de<br>accesos a memoria<br>(instrucciones más<br>operandos)           | load A                    | 2 + 1= 3        | <table><tr><td>mov F, A</td><td>3 + 2= 5</td></tr><tr><td>sub F, B</td><td>3 + 3= 6</td></tr><tr><td>mul F, C</td><td>3 + 3= 6</td></tr><tr><td>div D, E</td><td>3 + 3= 6</td></tr><tr><td>add F, D</td><td>3 + 3= 6</td></tr><tr><td><b>total</b></td><td><b>29 bytes</b></td></tr></table> |          | mov F, A                    | 3 + 2= 5 | sub F, B | 3 + 3= 6 | mul F, C | 3 + 3= 6 | div D, E | 3 + 3= 6 | add F, D | 3 + 3= 6 | <b>total</b> | <b>29 bytes</b> | sub F, A,<br>B | 4 + 3= 7 |
|  | mov F, A                  | 3 + 2= 5        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | sub F, B                  | 3 + 3= 6        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | mul F, C                  | 3 + 3= 6        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | div D, E                  | 3 + 3= 6        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | add F, D                  | 3 + 3= 6        |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | <b>total</b>              | <b>29 bytes</b> |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | sub B                     | 2 + 1= 3        | mul F, F,<br>C   | 4 + 3= 7 |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | mul C                     | 2 + 1= 3        | div D, D,<br>E   | 4 + 3= 7 |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
|  | store A                   | 2 + 1= 3        | add F, F,<br>D   | 4 + 3= 7 |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| load D   | 2 + 1= 3                  | <b>total</b>    | <b>28 bytes</b>  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| div E  | 2 + 1= 3                  |                 |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| add A  | 2 + 1= 3                  |                 |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| store F  | 2 + 1= 3                  |                 |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |
| <b>total</b>   | <b>24 bytes</b>           |                 |  |          |                             |          |          |          |          |          |          |          |          |          |              |                 |                |          |

(c) ¿Cuál máquina elegiría haciendo un balance de la cantidad de instrucciones, el espacio en memoria ocupado y el tiempo de ejecución (1 acceso a memoria= 1ms)? ¿Es ésta una conclusión general?

La máquina que elegiría haciendo un balance de la cantidad de instrucciones, el espacio en memoria ocupado y el tiempo de ejecución (1 acceso a memoria= 1ms) es la de 1 dirección. Sin embargo, ésta no es una conclusión general, ya que depende del tamaño del bus de datos respecto al bus de direcciones. Cuanto más grande es el primero respecto al segundo, más eficiente es migrar hacia máquinas de más direcciones.

**Ejercicio 4.**

*El siguiente programa utiliza una instrucción de transferencia de datos (instrucción MOV) con diferentes modos de direccionamiento para referenciar sus operandos. Ejecutar y analizar el funcionamiento de cada instrucción en el Simulador MSX88 observando el flujo de información a través del BUS DE DATOS, el BUS DE DIRECCIONES, el BUS DE CONTROL, el contenido de REGISTROS, de posiciones de MEMORIA, operaciones en la ALU, etc.*

```
ORG 1000h
NUM0 DB 0CAh
NUM1 DB 0
NUM2 DW ?
NUM3 DW 0ABCDh
NUM4 DW ?
END
```

```
ORG 2000H
MOV BL, NUM0
MOV BH, 0FFh
MOV CH, BL
MOV AX, BX
MOV NUM1, AL
MOV NUM2, 1234h
MOV BX, OFFSET NUM3
MOV DL, [BX]
MOV AX, [BX]
MOV BX, 1006h
MOV WORD PTR [BX], 0CDEFh
HLT
END
```

*(a) Explicar, detalladamente, qué hace cada instrucción MOV del programa anterior, en función de sus operandos y su modo de direccionamiento.*

MOV BL, NUM0: Con un modo de direccionamiento directo, copia el contenido de la variable NUM0 (CAh, 202 en decimal) a la parte baja del registro B.

MOV BH, 0FFh: Con un modo de direccionamiento inmediato, copia el valor FFh (255 en decimal) a la parte alta del registro B.

MOV CH, BL: Con un modo de direccionamiento por registro, copia el contenido de la parte baja del registro B (CAh, 202 en decimal) a la parte alta del registro C.

MOV AX, BX: Con un modo de direccionamiento por registro, copia el contenido del registro B (FFCAh, 65482 en decimal) al registro A.

MOV NUM1, AL: Con un modo de direccionamiento directo, copia el contenido de la parte baja del registro A (CAh, 202 en decimal) a la variable NUM1.



MOV NUM2, 1234h: Con un modo de direccionamiento inmediato, copia el valor 1234h (4660 en decimal) a la variable NUM2.

MOV BX, OFFSET NUM3: Con un modo de direccionamiento inmediato, copia la posición de memoria de NUM3 (1004h) al registro BX.

MOV DL, [BX]: Con un modo de direccionamiento indirecto por registro, copia el contenido que haya en la posición de memoria contenida en el registro B (CDh) a la parte baja del registro D.

MOV AX, [BX]: Con un modo de direccionamiento indirecto por registro, copia el contenido que haya en la posición de memoria contenida en el registro B (ABCDh) al registro A.

MOV BX, 1006h: Con un modo de direccionamiento inmediato, copia el valor 1006h (4102 en decimal) al registro B.

MOV WORD PTR [BX], 0CDEFh: Con un modo de direccionamiento indirecto por registro, copia el valor CDEFh (52719 en decimal) a la posición de memoria que apunta el registro B (1006h).

**(b)** Confeccionar una tabla que contenga todas las instrucciones MOV anteriores, el modo de direccionamiento y el contenido final del operando destino de cada una de ellas.

| Instrucción         | Modo de direccionamiento | AL | AH | BL | BH | CL | CH | DL | DH |
|---------------------|--------------------------|----|----|----|----|----|----|----|----|
| MOV BL, NUM0        | Directo                  |    |    | CA |    |    |    |    |    |
| MOV BH, 0FFh        | Inmediato                |    |    | CA | FF |    |    |    |    |
| MOV CH, BL          | Por registro             |    |    | CA | FF |    | CA |    |    |
| MOV AX, BX          | Por registro             | CA | FF | CA | FF |    | CA |    |    |
| MOV NUM1, AL        | Directo                  | CA | FF | CA | FF |    | CA |    |    |
| MOV NUM2, 1234h     | Inmediato                | CA | FF | CA | FF |    | CA |    |    |
| MOV BX, OFFSET NUM3 | Inmediato                | CA | FF | 04 | 10 |    | CA |    |    |
| MOV DL, [BX]        | Indirecto por registro   | CA | FF | 04 | 10 |    | CA | CD |    |
| MOV AX, [BX]        | Indirecto por registro   | CD | AB | 04 | 10 |    | CA | CD |    |
| MOV BX, 1006h       | Inmediato                | CD | AB | 06 | 10 |    | CA | CD |    |

|                                 |                           |    |    |    |    |  |    |    |  |
|---------------------------------|---------------------------|----|----|----|----|--|----|----|--|
| MOV WORD<br>PTR [BX],<br>0CDEFh | Indirecto por<br>registro | CD | AB | 06 | 10 |  | CA | CD |  |
|---------------------------------|---------------------------|----|----|----|----|--|----|----|--|

| Instrucción               | Modo de direccionamiento | NUM0<br>(1000) | NUM1<br>(1001) | NUM2<br>L<br>(1002) | NUM2<br>H<br>(1003) | NUM3<br>L<br>(1004) | NUM3<br>H<br>(1005) | NUM4<br>L<br>(1006) | NUM4<br>H<br>(1007) |
|---------------------------|--------------------------|----------------|----------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
| MOV BL, NUM0              | Directo                  | CA             | 00             |                     |                     | CD                  | AB                  |                     |                     |
| MOV BH, 0FFh              | Inmediato                | CA             | 00             |                     |                     | CD                  | AB                  |                     |                     |
| MOV CH, BL                | Por registro             | CA             | 00             |                     |                     | CD                  | AB                  |                     |                     |
| MOV AX, BX                | Por registro             | CA             | 00             |                     |                     | CD                  | AB                  |                     |                     |
| MOV NUM1, AL              | Directo                  | CA             | CA             |                     |                     | CD                  | AB                  |                     |                     |
| MOV NUM2, 1234h           | Inmediato                | CA             | CA             | 34                  | 12                  | CD                  | AB                  |                     |                     |
| MOV BX, OFFSET NUM3       | Inmediato                | CA             | CA             | 34                  | 12                  | CD                  | AB                  |                     |                     |
| MOV DL, [BX]              | Indirecto por registro   | CA             | CA             | 34                  | 12                  | CD                  | AB                  |                     |                     |
| MOV AX, [BX]              | Indirecto por registro   | CA             | CA             | 34                  | 12                  | CD                  | AB                  |                     |                     |
| MOV BX, 1006h             | Inmediato                | CA             | CA             | 34                  | 12                  | CD                  | AB                  |                     |                     |
| MOV WORD PTR [BX], 0CDEFh | Indirecto por registro   | CA             | CA             | 34                  | 12                  | CD                  | AB                  | EF                  | CD                  |

(c) Notar que durante la ejecución de algunas instrucciones MOV aparece en la pantalla del simulador un registro temporal denominado “ri”, en ocasiones acompañado por otro registro temporal denominado “id”. Explicar, con detalle, qué función cumplen estos registros.

El registro temporal denominado “ri” cumple la función de guardar, temporalmente, la dirección de la variable (fuente o destino, dependiendo del caso). El registro temporal denominado “id” cumple la función de guardar temporalmente el valor que se quiere copiar en la variable.

**Ejercicio 5.**

*El siguiente programa utiliza diferentes instrucciones de procesamiento de datos (instrucciones aritméticas y lógicas). Analizar el comportamiento de ellas y ejecutar el programa en el MSX88.*

```
ORG 1000H
NUM0 DB 80h
NUM1 DB 200
NUM2 DB -1
BYTE0 DB 01111111B
BYTE1 DB 10101010B
```

```
ORG 2000H
MOV AL, NUM0
ADD AL, AL
INC NUM1
MOV BH, NUM1
MOV BL, BH
DEC BL
SUB BL, BH
MOV CH, BYTE1
AND CH, BYTE0
NOT BYTE0
OR CH, BYTE0
XOR CH, 11111111B
HLT
END
```

**(a)** *¿Cuál es el estado de los FLAGS después de la ejecución de las instrucciones ADD y SUB del programa anterior? Justificar el estado (1 o 0) de cada uno de ellos. ¿Dan alguna indicación acerca de la correctitud de los resultados?*

En la operación ADD, se suma 80h (128 en decimal) más 80h (128 en decimal), lo que debería dar 100h (256 en decimal). El estado de los flags después de la ejecución de la instrucción ADD es Z= 1 (resultado cero), N= 0 (resultado no negativo), C= 1 (resultado incorrecto en BSS), V= 1 (resultado incorrecto en Ca2).

En la operación SUB, se resta 200 menos 201, lo que debería dar -1. El estado de los flags después de la ejecución de la instrucción SUB es Z= 0 (resultado no cero), N= 1 (resultado negativo), C= 1 (resultado incorrecto en BSS), V= 0 (resultado correcto en Ca2).

La correctitud de las operaciones ADD y SUB depende:

- en BSS, de la bandera C (carry); si se tiene C= 1, el resultado va a ser incorrecto; y
- en Ca2, de la bandera V (overflow); si se tiene V= 1, el resultado va a ser incorrecto.

**(b)** ¿Qué cadenas binarias representan a NUM1 y NUM2 en la memoria del simulador?  
¿En qué sistemas binarios están expresados estos valores?

Las cadenas binarias que representan NUM1 y NUM2 en la memoria del simulador (al finalizar la ejecución del programa) son 11001001b - C9h (201 en decimal) y 11111111b - FFh (-1 en decimal), respectivamente. Estos valores están expresados en BSS y Ca2, respectivamente.

**(c)** Confeccionar una tabla que indique para cada operación aritmética o lógica del programa, el valor de sus operandos, en qué registro o dirección de memoria se almacenan y el resultado de cada operación.

| Operación         | Operando 1      | Operando 2      | Dirección 1 | Dirección 2 | Resultado              |
|-------------------|-----------------|-----------------|-------------|-------------|------------------------|
| ADD AL, AL        | 80h             | 80h             | AL          | AL          | 00h (Z= 1, C= 1, V= 1) |
| INC NUM1          | 200 (C8h)       |                 | 1001h       |             | 201 (C9h)              |
| DEC BL            | 201 (C9h)       |                 | BL          |             | 200 (C8h)              |
| SUB BL, BH        | 200 (C8h)       | 201 (C9h)       | BL          | BH          | -1 (FF) (N= 1, C= 1)   |
| AND CH, BYTE0     | 10101010b (AAh) | 01111111b (7Fh) | CH          | 1003h       | 00101010b (2Ah)        |
| NOT BYTE0         | 01111111b (7Fh) |                 | 1003h       |             | 10000000b (80h)        |
| OR CH, BYTE0      | 00101010b (2Ah) | 10000000b (80h) | CH          | 1003h       | 10101010b (AAh)        |
| XOR CH, 11111111B | 10101010b (AAh) | 11111111b (FFh) | CH          |             | 01010101b (55h)        |

**Ejercicio 6.**

*El siguiente programa implementa un contador utilizando una instrucción de transferencia de control. Analizar el funcionamiento de cada instrucción y, en particular, las del lazo repetitivo que provoca la cuenta.*

```
ORG 1000H
INI DB 0
FIN DB 15

ORG 2000H
MOV AL, INI
MOV AH, FIN
SUMA: INC AL
      CMP AL, AH
      JNZ SUM
      HLT
      END
```

**(a)** *¿Cuántas veces se ejecuta el lazo? ¿De qué variables depende esto en el caso general?*

El lazo se ejecuta 15 veces. En el caso general, esto depende de las variables INI y FIN.

**(b)** *Analizar y ejecutar el programa reemplazando la instrucción de salto condicional JNZ por las siguientes, indicando, en cada caso, el contenido final del registro AL:*

- JS: 15.
- JZ: 1.
- JMP: no tiene contenido final (lazo infinito).

**Ejercicio 7.**

*Escribir un programa en lenguaje Assembly del MSX88 que implemente la sentencia condicional de un lenguaje de alto nivel  $IF A < B THEN C = A ELSE C = B$ . Considerar que las variables de la sentencia están almacenadas en los registros internos de la CPU del siguiente modo  $A$  en  $AL$ ,  $B$  en  $BL$  y  $C$  en  $CL$ . Determinar las modificaciones que debería hacer al programa si la condición de la sentencia  $IF$  fuera:*

```

                                ORG 2000H
                                CMP AL, BL
                                JS IF
                                MOV CL, BL
                                JMP FIN
IF:                             MOV CL, AL
FIN:                             HLT
                                END

```

(a)  $A \leq B$ .

Opción 1:

```

                                ORG 2000H
                                CMP AL, BL
                                JS IF
                                JZ IF
                                MOV CL, BL
                                JMP FIN
IF:                             MOV CL, AL
FIN:                             HLT
                                END

```

Opción 2:

```

                                ORG 2000H
                                CMP AL, BL
                                JNS AUX
AUX:                             JNZ ELSE
                                MOV CL, AL
                                JMP FIN
ELSE:                             MOV CL, BL
FIN:                             HLT
                                END

```

(b)  $A = B$ .

Opción 1:

```
ORG 2000H
CMP AL, BL
JZ IF
MOV CL, BL
JMP FIN
IF:    MOV CL, AL
FIN:   HLT
      END
```

Opción 2:

```
ORG 2000H
CMP AL, BL
JNZ ELSE
MOV CL, AL
JMP FIN
ELSE:  MOV CL, BL
FIN:   HLT
      END
```



**Ejercicio 8.**

*El siguiente programa suma todos los elementos de una tabla almacenada a partir de la dirección 1000H de la memoria del simulador. Analizar el funcionamiento y determinar el resultado de la suma. Comprobar resultado en el MSX88.*

```
ORG 1000H
TABLA DB 2,4,6,8,10,12,14,16,18,20
FIN DB ?
TOTAL DB ?
MAX DB 13

ORG 2000H
MOV AL, 0
MOV CL, OFFSET FIN - OFFSET TABLA
MOV BX, OFFSET TABLA
SUMA: ADD AL, [BX]
      INC BX
      DEC CL
      JNZ SUMA
      HLT
      END
```

*¿Qué modificaciones se deberá hacer en el programa para que el mismo almacene el resultado de la suma en la celda etiquetada TOTAL?*

El resultado de la suma es 110.

```
ORG 1000H
TABLA DB 2,4,6,8,10,12,14,16,18,20
FIN DB ?
TOTAL DB ?
MAX DB 13

ORG 2000H
MOV AL, 0
MOV CL, OFFSET FIN - OFFSET TABLA
MOV BX, OFFSET TABLA
SUMA: ADD AL, [BX]
      INC BX
      DEC CL
      JNZ SUMA
      MOV TOTAL, AL
      HLT
      END
```

**Ejercicio 9.**

*Escribir un programa que, utilizando las mismas variables y datos que el programa del punto anterior (TABLA, FIN, TOTAL, MAX), determine cuántos de los elementos de TABLA son menores o iguales que MAX. Dicha cantidad debe almacenarse en la celda TOTAL.*

```
ORG 1000H
TABLA DB 2,4,6,8,10,12,14,16,18,20
FIN DB ?
TOTAL DB ?
MAX DB 13

ORG 2000H
MOV AL, 0
MOV AH, MAX
MOV CL, OFFSET FIN - OFFSET TABLA
MOV BX, OFFSET TABLA
LAZO:  CMP AH, [BX]
      INC BX
      JS MAYOR
      INC AL
MAYOR: DEC CL
      JNZ LAZO
      MOV TOTAL, AL
      HLT
      END
```

**Ejercicio 10.**

*Analizar el funcionamiento del siguiente programa:*

```
CARGA:  ORG 2000H
        MOV AX, 1
        MOV BX, 1000h
        MOV [BX], AX
        ADD BX, 2
        ADD AX, AX
        CMP AX, 200
        JS CARGA
        HLT
        END
```

**(a)** *El programa genera una tabla. ¿Cómo están relacionados sus elementos entre sí?*

Los elementos entre sí están relacionados de manera tal que cada valor siguiente será el doble del valor anterior, comenzando la tabla en el valor 1.

**(b)** *¿A partir de qué dirección de memoria se crea la tabla? ¿Cuál es la longitud de cada uno de sus elementos (medida en bits)?*

La tabla se crea a partir de la dirección de memoria 1000h (4096 en decimal) y la longitud de cada uno de sus elementos es de 16 bits (DW).

**(c)** *¿Cuántos elementos tiene la tabla una vez finalizada la ejecución del programa? ¿De qué depende esta cantidad?*

La tabla, una vez finalizada la ejecución del programa, tiene 8 elementos (1, 2, 4, 8, 16, 32, 64, 128).

**Ejercicio 11.**

*Escribir un programa que genere una tabla a partir de la dirección de memoria almacenada en la celda DIR con los múltiplos de 5 desde cero hasta MAX.*

```
                ORG 1000H
                MAX DB 50
                DIR DB ?

                ORG 2000H
                MOV AL, 0
                MOV AH, MAX
                MOV BX, OFFSET DIR
MULT:           MOV [BX], AL
                INC BX
                ADD AL, 5
                CMP AL, AH
                JS MULT
                HLT
                END
```

**Ejercicio 12.**

*Escribir un programa que, dado un número X, genere un arreglo con todos los resultados que se obtienen hasta llegar a 0, aplicando la siguiente fórmula: si X es par, se le resta 7; si es impar, se le suma 5 y al resultado se le aplica, nuevamente, la misma fórmula. Por ejemplo: si X= 3, entonces, el arreglo tendrá 8, 1, 6, -1, 4, -3, 2, -5, 0.*

**Opción 1:**

```

                                ORG 1000H
                                NUM DB 20
                                TABLA DB ?

                                ORG 2000H
                                MOV AL, NUM
                                MOV BX, OFFSET TABLA - 2
                                INC BX
LAZO:                          JZ FIN
                                INC BX
                                MOV AH, AL
                                AND AH, 1
                                JZ PAR
                                ADD AL, 5
                                MOV [BX], AL
                                JMP LAZO
PAR:                           SUB AL, 7
                                MOV [BX], AL
                                JMP LAZO
FIN:                           HLT
                                END

                                ORG 1000H
                                NUM DB 20
                                TABLA DB ?
                                END

```

**Opción 2:**

```

                                ORG 1000H
                                NUM DB 20
                                TABLA DB ?

                                ORG 2000H
                                MOV AL, NUM
                                MOV BX, OFFSET TABLA - 2
                                INC BX
LAZO:                          JZ FIN
                                INC BX
                                MOV AH, AL

```

```

                AND AH, 1
                JNZ IMPAR
                SUB AL, 7
                MOV [BX], AL
                JMP LAZO
IMPAR:          ADD AL, 5
                MOV [BX], AL
                JMP LAZO
FIN:            HLT
                END
```

**Ejercicio 13.**

*Dada la frase “Organización y la Computación”, almacenada en la memoria, escribir un programa que determine cuántas letras “a” seguidas de “c” hay en ella.*

```
ORG 1000H
AC DB ?
INICIO DB “Organización y la Computación”
FINAL DB ?

ORG 2000H
MOV AL, 0
MOV BX, OFFSET INICIO - 1
MOV CX, OFFSET FINAL
LAZO: INC BX
      CMP CX, BX
      JZ FIN
      MOV DX, [BX]
      CMP DX, 9799h
      JZ SUMA
      JMP LAZO
SUMA: INC AL
      JMP LAZO
FIN:  MOV AC, AL
      HLT
      END
```

**Ejercicio 14.**

*Escribir un programa que sume dos números representados en Ca2 de 32 bits almacenados en memoria de datos y etiquetados NUM1 y NUM2 y guarde el resultado en RESUL (en este caso, cada dato y el resultado ocuparán 4 celdas consecutivas de memoria). Verificar el resultado final y almacenar 0FFH en la celda BIEN, en caso de ser correcto, o en otra MAL, en caso de no serlo. Recordar que el MSX88 trabaja con números en Ca2, pero tener en cuenta que las operaciones con los 16 bits menos significativos de cada número deben realizarse en BSS.*

```
ORG 1000H
NUM1 DW 0AAFFH, 0BBFFH
NUM2 DW 0H, 1
BIEN DB ?
MAL DB ?
RESUL DW ?

ORG 2000H
MOV BX, OFFSET NUM1 + 2
MOV AX, [BX]
MOV BX, OFFSET NUM2 + 2
ADD AX, [BX]
MOV BX, OFFSET RESULT + 2
MOV [BX], AX
MOV AX, NUM1
ADC AX, NUM2
MOV RESUL, AX
JO M
JC M
MOV BIEN, 0FFH
JMP FIN
M: MOV MAL, 0FFH
FIN: HLT
END
```



**Ejercicio 15.**

*Escribir un programa que efectúe la suma de dos vectores de 6 elementos cada uno (donde cada elemento es un número de 32 bits) almacenados en memoria de datos y etiquetados TAB1 y TAB2 y guarde el resultado en TAB3. Suponer, en primera instancia, que no existirán errores de tipo aritmético (ni carry ni overflow), luego analizar y definir los cambios y agregados necesarios que deberían realizarse al programa para tenerlos en cuenta.*

Sin errores de tipo aritmético:

```

ORG 1000H
TAB1 DW 1,2,3,4,5,6,7,8,9,10,11,12
TAB2 DW 13,14,15,16,17,18,19,20,21,22,23,24
TAB3 DW 12 DUP (?)
CANT DB 12
DIR3 DW ?

ORG 2000H
MOV AX, OFFSET TAB1
MOV CX, OFFSET TAB2
MOV DIR3, OFFSET TAB3
LAZO: MOV BX, AX
      MOV DX, [BX]
      MOV BX, CX
      ADD DX, [BX]
      MOV BX, DIR3
      MOV [BX], DX
      ADD AX, 2
      ADD CX, 2
      ADD DIR3, 2
      DEC CANT
      JNZ LAZO
      HLT
      END

```

Con errores de tipo aritmético:

```

ORG 1000H
TAB1 DW 1,2,3,4,5,6,7,8,9,10,11,12
TAB2 DW 13,14,15,16,17,18,19,20,21,22,23,24
TAB3 DW 12 DUP (?)
CANT DB 6
DIR3 DW ?

ORG 2000H
MOV AX, OFFSET TAB1
MOV CX, OFFSET TAB2
MOV DIR3, OFFSET TAB3

```

LAZO:       MOV BX, AX  
              MOV DX, [BX]  
              MOV BX, CX  
              ADD DX, [BX]  
              PUSHF  
              MOV BX, DIR3  
              MOV [BX], DX  
              ADD AX, 2  
              ADD CX, 2  
              ADD DIR3, 2  
              MOV BX, AX  
              MOV DX, [BX]  
              MOV BX, CX  
              POPF  
              ADC DX, [BX]  
              MOV BX, DIR3  
              MOV [BX], DX  
              ADD AX, 2  
              ADD CX, 2  
              ADD DIR3, 2  
              DEC CANT  
              JNZ LAZO  
              HLT  
              END

**Ejercicio 16.**

Los siguientes programas realizan la misma tarea, en uno de ellos se utiliza una instrucción de transferencia de control con retorno. Analizar y comprobar la equivalencia funcional.

**Programa 1:**

```
ORG 1000H
NUM1 DB 5H
NUM2 DB 3H

ORG 2000H
MOV AL, NUM1
CMP AL, 0
JZ FIN
MOV AH, 0
MOV DX, 0
MOV CL, NUM2
LOOP:  CMP CL, 0
        JZ FIN
        ADD DX, AX
        DEC CL
        JMP LOOP
FIN:    HLT
        END
```

**Programa 2:**

```
ORG 1000H
NUM1 DB 5H
NUM2 DB 3H

ORG 3000H
SUB1:  CMP AL, 0
        JZ FIN
        CMP CL, 0
        JZ FIN
        MOV AH, 0
        MOV DX, 0
LAZO:  ADD DX, AX
        DEC CX
        JNZ LAZO
FIN:    RET

ORG 2000H
MOV AL, NUM1
MOV CL, NUM2
CALL SUB1
HLT
```

END

(a) *¿Cuál es la tarea realizada por ambos programas?*

La tarea realizada por ambos programas es multiplicar NUM1 por NUM2.

(b) *¿Dónde queda almacenado el resultado?*

El resultado queda almacenado en el registro DX.

(c) *¿Cuál programa realiza la tarea más rápido? ¿El tiempo de ejecución de la tarea depende de los valores almacenados en NUM1, en NUM2, en ambos lugares o en ninguno?*

El programa que realiza la tarea más rápido es el primero, ya que hace menos llamados a memoria que el segundo. Por otra parte, el tiempo de ejecución de la tarea depende de los valores almacenados en NUM2.

(d) *Explicar, detalladamente, todas las acciones que tienen lugar al ejecutarse la instrucción CALL SUB1.*

Al ejecutarse la instrucción CALL SUB1, se guarda el valor de la posición de memoria que está en el puntero de instrucción (IP) en la pila (PUSH del IP), se asigna el valor de la posición de memoria correspondiente a la etiqueta SUB1 al IP y la CPU comienza a ejecutar las instrucciones de la subrutina SUB1.

(e) *¿Qué operación se realiza con la instrucción RET? ¿Cómo sabe la CPU a qué dirección de memoria debe retornar desde la subrutina al programa principal?*

La operación que se realiza con la instrucción RET es retornar al programa principal a partir de la instrucción siguiente a la instrucción CALL SUB1. La CPU sabe a qué dirección de memoria debe retornar desde la subrutina al programa principal porque el puntero de instrucción (IP) se carga con el valor de la posición de memoria guardada en la pila (POP del IP) y, por lo tanto, la ejecución del programa sigue a partir de la instrucción siguiente a la instrucción CALL SUB1.

**Ejercicio 17.**

*El siguiente programa es otra forma de implementación de la tarea del punto anterior (Ejercicio 16). Analizar y establecer las diferencias con las anteriores, en particular las relacionadas a la forma de “proveer” los operandos a las subrutinas. Explicar detalladamente:*

```
ORG 1000H
NUM1 DW 5H
NUM2 DW 3H

SUB2:  ORG 3000H
LAZO:  MOV DX, 0
        MOV BX, AX
        ADD DX, [BX]
        PUSH DX
        MOV BX, CX
        MOV DX, [BX]
        DEC DX
        MOV [BX], DX
        POP DX
        JNZ LAZO
        RET

ORG 2000H
MOV AX, OFFSET NUM1
MOV CX, OFFSET NUM2
CALL SUB2
HLT
END
```

**(a)** *Todas las acciones que tienen lugar al ejecutarse las instrucciones PUSH DX y POP DX.*

Al ejecutarse la instrucción PUSH DX, se resta dos al valor apuntado en el registro SP y se guarda el valor de DX en la pila. Al ejecutarse la instrucción POP DX, se suma dos al valor apuntado en el registro SP y se carga el valor almacenado en la pila en DX.

**(b)** *Cuáles son los dos usos que tiene el registro DX en la subrutina SUB2.*

Los dos usos que tiene el registro DX en la subrutina SUB2 son de acumulador y de decrementador.

**Ejercicio 18.**

*Escribir un programa que sume 2 vectores de 6 elementos (similar al realizado en el Ejercicio 15), de modo tal que utilice una subrutina que sume números de 32 bits (similar al programa escrito en el Ejercicio 14).*

```

                                ORG 1000H
                                TAB1 DW 1,2,3,4,5,6,7,8,9,10,11,12
                                TAB2 DW 13,14,15,16,17,18,19,20,21,22,23,24
                                TAB3 DW 12 DUP (?)
                                CANT DB 6
                                DIR3 DW ?

SUMA:                          ORG 3000H
                                MOV BX, AX
                                MOV DX, [BX]
                                MOV BX, CX
                                ADD DX, [BX]
                                PUSHF
                                MOV BX, DIR3
                                MOV [BX], DX
                                ADD AX, 2
                                ADD CX, 2
                                ADD DIR3, 2
                                MOV [BX], AX
                                MOV DX, [BX]
                                MOV BX, CX
                                POPF
                                ADC DX, [BX]
                                MOV BX, DIR3
                                MOV [BX], DX
                                ADD AX, 2
                                ADD CX, 2
                                ADD DIR3, 2
                                DEC CANT
                                JNZ SUMA
                                RET

                                ORG 2000H
                                MOV AX, OFFSET TAB1
                                MOV CX, OFFSET TAB2
                                MOV DIR3, OFFSET TAB3
                                CALL SUMA
                                HLT
                                END
```

**Ejercicio 19.**

*Escribir una subrutina que reciba la mantisa entera en BSS y el exponente en BSS de un número en los registros AH y AL, respectivamente, y devuelva, en ellos, una representación equivalente del mismo pero con el exponente disminuido en 1 y la mantisa ajustada. De no ser posible el ajuste, BL debe contener 0FFH en vez de 00H en el retorno.*

```
                ORG 1000H
                MANTISA DB 0AAh
                EXPONENTE DB 5

                ORG 3000H
AJUSTAR:        CMP AL, 0
                JZ NOPOSIBLE
                DEC AL
                ADD AH, AH
                JC NOAJUSTA
                MOV BL, 00H
                JMP FIN
NOPOSIBLE:      MOV BL, 0FFH
FIN:            RET

                ORG 2000H
                MOV AH, MANTISA
                MOV AL, EXPONENTE
                CALL AJUSTAR
                HLT
                END
```

**Ejercicio 20.**

*Escribir una subrutina que reciba como parámetro un número en el formato IEEE 754 de simple precisión y analice/verifique las características del mismo devolviendo en el registro CL un valor igual a 0 si el número está sin normalizar, 1 en caso de ser +/- infinito, 2 si es un NAN, 3 si es un +/- cero y 4 si es un número normalizado. La subrutina recibe en AX la parte alta del número y en BX la parte baja.*

```

                                ORG 1000H
                                MANTISA DB 0AAH, 0BBH, 0CCH
                                EXPONENTE DB 00AH

ANALIZAMANTISA:                ORG 3000H
                                MOV DH, [BX]
                                CMP DH, 128
                                JNZ MANTISA_NOCERO
LAZO:                          MOV DH, [BX]
                                CMP DH, 0
                                JNZ MANTISA_NOCERO
                                INC BX
                                CMP BX, OFFSET EXPONENTE
                                JZ MANTISA_CERO
                                JMP LAZO
MANTISA_CERO:                  MOV DL, 0
                                JMP FIN_MANTISA
MANTISA_NOCERO:                MOV DL, 1
FIN_MANTISA:                   RET

NORMALIZADO:                   CALL ANALIZAMANTISA
                                CMP AL, 0
                                JZ SINNORMALIZAR
                                CMP AL, 255
                                JZ NAN
                                MOV CL, 4
                                JMP FIN_EXPONENTE
SINNORMALIZAR:                 CMP DL, 0
                                JZ CERO
                                MOV CL, 0
                                JMP FIN_EXPONENTE
INFINITO:                      MOV CL, 1
                                JMP FIN_EXPONENTE
NAN:                           CMP DL, 0
                                JZ INFINITO
                                MOV CL, 2
                                JMP FIN_EXPONENTE
CERO:                          MOV CL, 3
FIN_EXPONENTE:                 RET

                                ORG 2000H

```



```
MOV BX, OFFSET MANTISA  
MOV AL, EXPONENTE  
CALL NORMALIZADO  
HLT  
END
```

**Ejercicio 21.**

*Modificar la subrutina del Ejercicio 19 para el caso en que la mantisa y el exponente estén representados en BCS.*

```
                                ORG 1000H
                                MANTISA DB 0AAh
                                EXPONENTE DB 5

AJUSTAR:                       ORG 3000H
                                DEC AL
                                ADD AH, AH
                                JC NOPOSIBLE1
                                JO NOPOSIBLE2
POSIBLE:                       MOV BL, 00H
                                JMP FIN
NOPOSIBLE1:                   JO POSIBLE
                                MOV BL, 0FFH
                                JMP FIN
NOPOSIBLE2:                   MOV BL, 0FFH
FIN:                           RET

                                ORG 2000H
                                MOV AH, MANTISA
                                MOV AL, EXPONENTE
                                CALL AJUSTAR
                                HLT
                                END
```