Organización de Computadoras 2023

Práctica 1

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	

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
   01001111   □ Acarreos
+  01111000
   11000111
```

Interpretación en BSS

```
    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
   01001111   □Acarreos
+  01111000
   11000111
```

BCS	Ca1	Ca2	Exceso	Observe los flags!
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?

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 descriptas en sistema Ca2. ¿Qué pasa en el caso (e)?
    a. 00100110 b. 11011000 c. 00111000 d. 00000000 e. 10000000

18. Interprete las siguientes cadenas descriptas en sistema $Ex2^{(n-1)}$ con n=8. ¿Qué pasa en el caso (e)?
    a. 10100110 b. 01011000 c. 10111000 d. 10000000 e. 00000000

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	
1010000010000	
1110001011101	
111010011001011	
1001111100100011	
1110101011001010	
101101101011010	

Hexadecimal a BCH	
2801	
1C5D	
78AB	
F79A	
7EF1	
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 +00011011					
01110000 +11110001					
00011101 -00011011					
01110000 -11110001					
10011101 +01110010					

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

00111010 -00001111					
00000000 -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 +00011011					
01110000 +11110001					
00011101 -00011011					
01110000 -11110001					
10011101 +01110010					
01001100 +01110000					
10011101 -01110010					

01001100 -01110000					
01110110 +01110001					
11001100 +11110000					
01110110 -01110001					
11001100 -11110000					
10111001 +11100011					
10000000 +10000000					
10111001 -11100011					
10000000 -10000000					
00111010 +00001111					
00000000 +10000000					
00111010 -00001111					
00000000 -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. Recuerde que la coma es solo ilustrativa y no forma parte de las cadenas binarias.

```
 000111,01
+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 Ca2 y Ex2^{n-1} (con n=8). ¿Qué pasa en el último caso?

Binario	Decimal (interpretado en Ca2)	Decimal (interpretado en Ex2^{n-1})
00100110		
11011000		
00111000		
00000000		
10000000		

Organización de Computadoras 2023

Práctica 2

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:

a. Mantisa fraccionaria en BSS de 8 bits y exponente en BSS 4 bits
b. Mantisa fraccionaria normalizada en BSS de 15 bits y exponente en CA1 10 bits
c. Mantisa fraccionaria normalizada con bit implícito en BCS de 15 bits y exponente en Exceso 5 bits
d. 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			

5. Diga como influyen las siguientes variantes en el rango y resolución:
   a. Mantisa con signo y sin signo.
   b. Exponente con signo y sin signo.
   c. Tamaño de mantisa.
   d. Tamaño de exponente.
   e. Mantisa fraccionaria, fraccionaria normalizada y fraccionaria normalizada con bit implícito.

6. Efectúe las siguientes sumas para un sistema de punto flotante con mantisa BSS de 8 bits y exponente en BCS 8 bits.

00001111 00000011 + 00001000 00000010 = ¿?
01111111 00000000 + 11111100 10000001 = ¿?
00000001 00000111 + 00011100 00000000 = ¿?

Observe que los factores de escala deben ser los mismos, sino sumaríamos 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).

7. Suponiendo que los números que no son representables se aproximan al más próximo, obtenga las representaciones o aproximaciones de los números 8,625; 0,4 y 2,5 en los sistemas:
   a. Mantisa fraccionaria normalizada de 5 bits BSS exponente 4 bits CA2
   b. Mantisa fraccionaria normalizada de 10 bits BCS exponente 3 bits CA2

8. Definimos Error Absoluto y Error Relativo de un número **x** en un sistema de la siguiente forma:
   $EA(x) = | x' - x |$ y $ER(x) = EA(x) / x$; donde x' es el número representable del sistema más próximo a x.
   Calcule los errores absolutos y relativos para los casos del ejercicio anterior.

9. Considerando que en los procesos de truncamiento o redondeo la elección se basa en la representación más cercana, estime el Error Absoluto Máximo cometido en las representaciones del ejercicio 1. Recuerde que la distancia entre 2 representaciones sucesivas se conoce como resolución (R), por lo que $EAmáx \leq R / 2$.

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

11. Detalle las características del estándar IEEE 754 para simple precisión y doble precisión

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

0 11000100 00000000000000000000000
1 11111110 10100000000000000000000
0 00000000 00000000000000000000001
0 00000000 10011000000000000000000
1 00000000 00000000000000000000000
0 11111111 00000000000000000000000
0 11111111 00000100000000000000000
0 01100010100 00000000000000000000000000000000000000000000000000000
0 11010101110 10100000000000001000000000000000000000000000000000000
0 00000000000 01010000000000000000000000000000000000000000000000000
1 11111111111 11111000000000000000000000000000000000000000000000000

13. Hallar la representación en simple precisión del estándar IEEE 754 de los siguientes números
1; 13; 257; -40000; 0,0625

14. Calcule 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?

15. Efectúe las siguientes sumas (las cadenas son representaciones en el estándar IEEE 754)

00001111 01000000000000000000000 + 00010000 01000000000000000000000 = ¿?
11111111 10101010101010101010101 + 11111100 10000001111100000110101 0 =¿?

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?

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:

a. 10011001 **AND** 10101110
b. 01011000 **AND** 11110011
c. 10011001 **OR** 10101110
d. 01011000 **OR** 11110011
e. 10011001 **XOR** 10101110
f. 01011000 **XOR** 11110011
g. **NOT** 010111000
h. **NOT** 111010100
i. 10011001 **NAND** 10101110
j. 01011000 **NAND** 11110011
k. 10111001 **NOR** 11101110
l. 01011010 **NOR** 11010011
m. 10111001 **XNOR** 11101110
n. 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 mascara?**
En los casos de más de una operación, obtenga el resultado y a ese resultado aplíquele la operación siguiente.Ejemplo:

$X_7X_6X_5X_4X_3X_2X_1X_0$
AND $0_71_60_50_40_31_2\ 0_10_0$

$0_7X_60_50_40_3X_20_10_0$
XOR $0_71_60_50_40_30_21_10_0$

$0_7{\sim}X_60_50_40_3X_21_10_0$ ***Nota: ~X = X negado (valor opuesto a X)***

a. **X OR 00011000**
b. **X OR 11001100**
c. **X AND 01010101**
d. **X AND 01001100**
e. **X XOR 01010101**
f. **X XOR 11001100**

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.

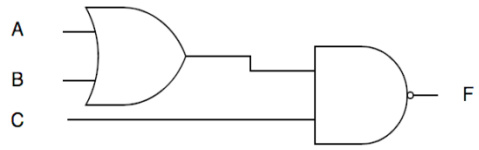
4. Dado un byte **X=[X_7,X_6,X_5,X_4,X_3,X_2,X_1,X_0]** (los X representan bits con valores indeterminados), aplíquele operaciones lógicas (1 o más) con un byte MASK, que deberá también determinar, para lograr los siguientes efectos:

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 6a 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.

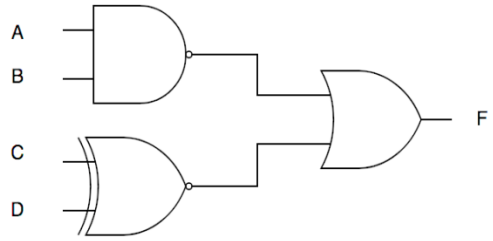
Circuitos Combinatorios

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

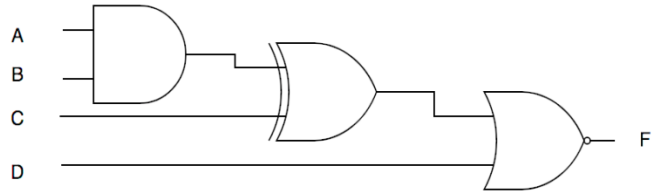
1)



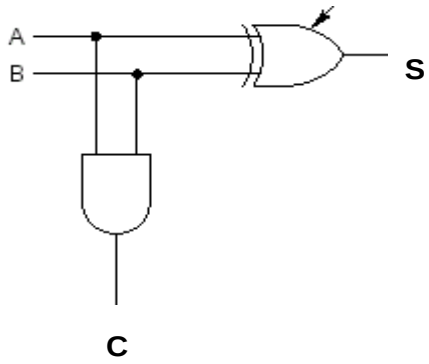
2)



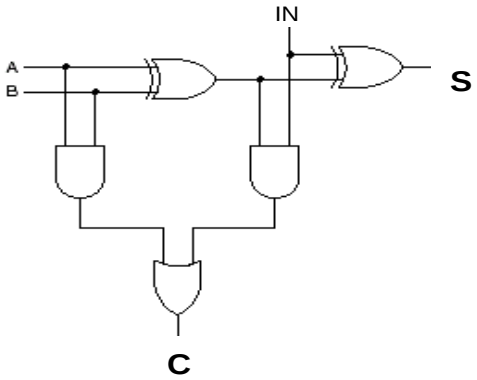
3)



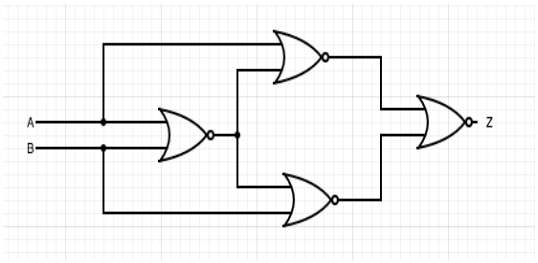
4)



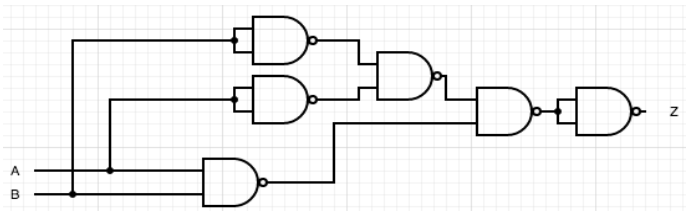
5)



6)



7)



Organización de Computadoras 2023

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.***

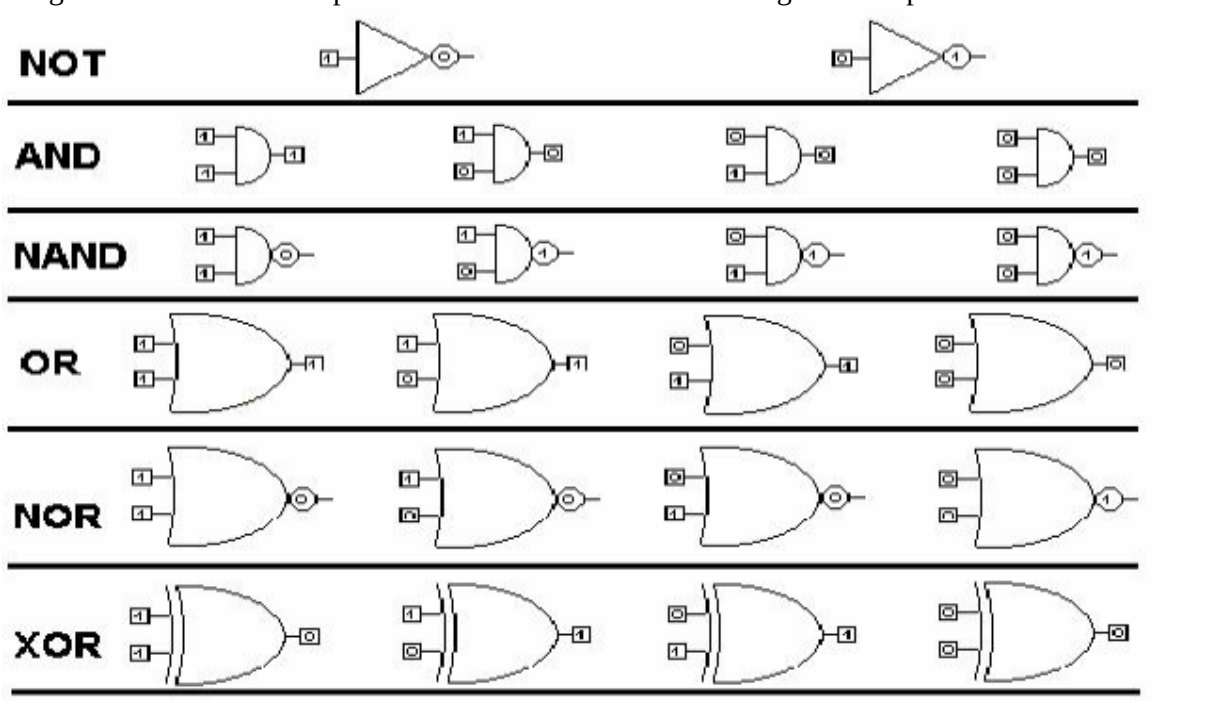
Bibliografía:

- ***"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".***

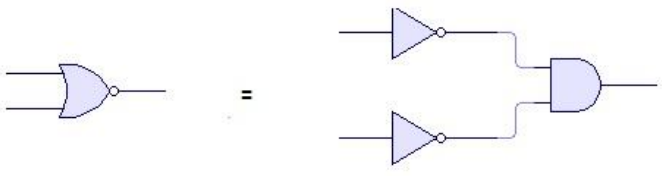
Tener en cuenta para resolución de ejercicios 6 al 10:

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)



$\overline{A+B}$

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

$\overline{A} \cdot \overline{B}$

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

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:

- $\overline{A} \cong \overline{A} + \overline{A} \cong \overline{A.A}$ (Aplico 2 equivalencias lógicas, la última es la ley de De Morgan).
- $A + B \cong \overline{\overline{A + B}} \cong \overline{\overline{A}.\overline{B}} \cong \overline{(\overline{A.A})}\overline{(\overline{B.B})}$ (doble negación, De Morgan, equivalencia anterior para la negación).
- $A.B \cong \overline{\overline{A.B}} \cong \overline{(\overline{A.B})}\overline{(\overline{A.B})}$ (doble negación, 1er equivalencia para la negación).
- $A \otimes B \cong (A.\overline{B}) + (\overline{A}.B)$..... (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:

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

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

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

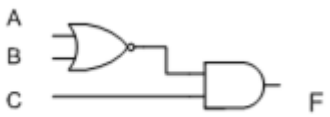
d) $A + A + B = A + B + B$

e) $A + B.C = A.C + B$

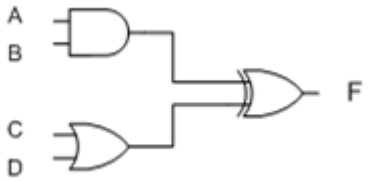
f) $A \oplus B = A \oplus B$

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

1)

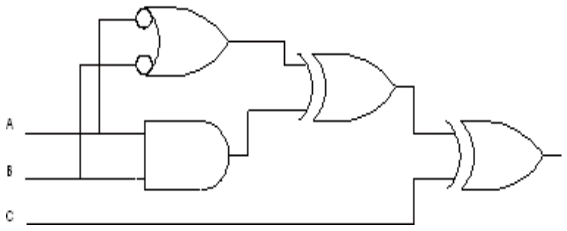


2)



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.



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

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, ∨, +) 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)

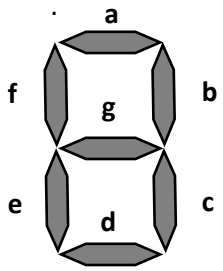
Rta: cuando viene A=0 **y** B=1 **y** C=0 **y** D=1. O dicho de otra manera, cuando NO ocurra A y SI ocurra B y NO ocurra C y SI ocurra D.
Conclusión: Se puede representar una ecuación que retorne 1 cuando en las cuatro entradas reciba el número 5, de la siguiente manera: $F_5 = \overline{A}.B.\overline{C}.D$ (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).

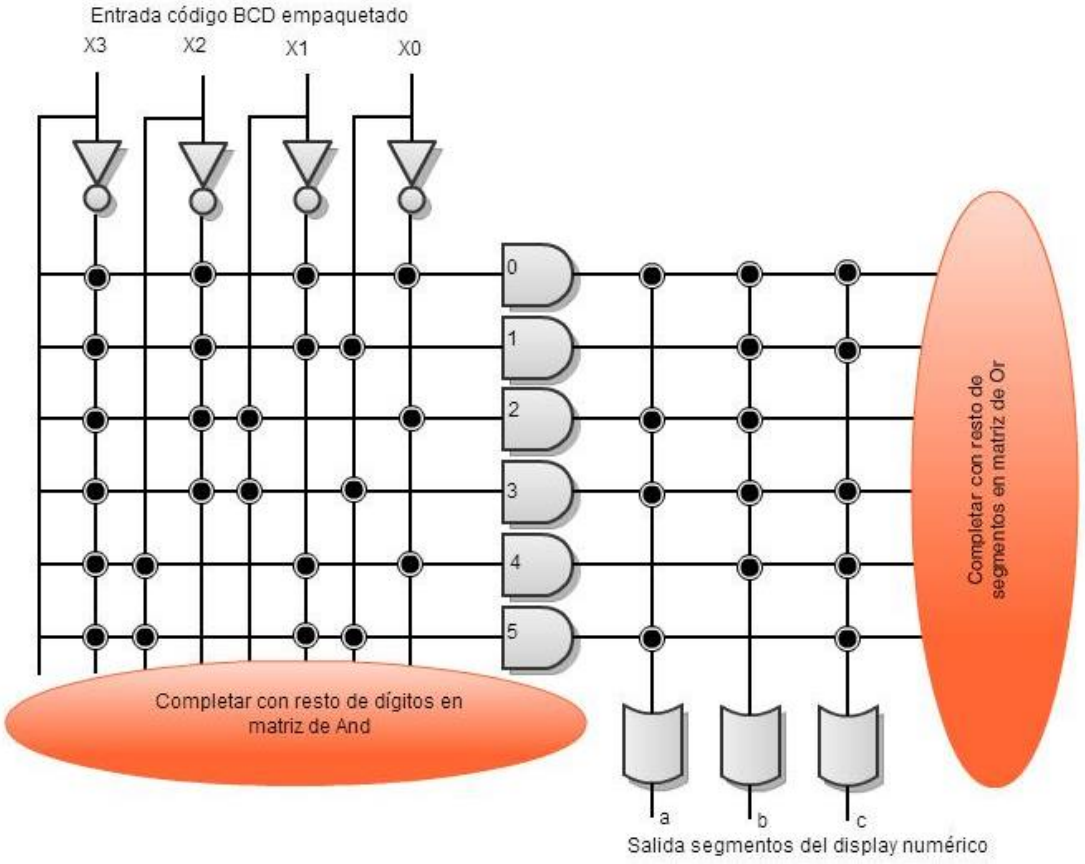
A	B	C	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**

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.

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.

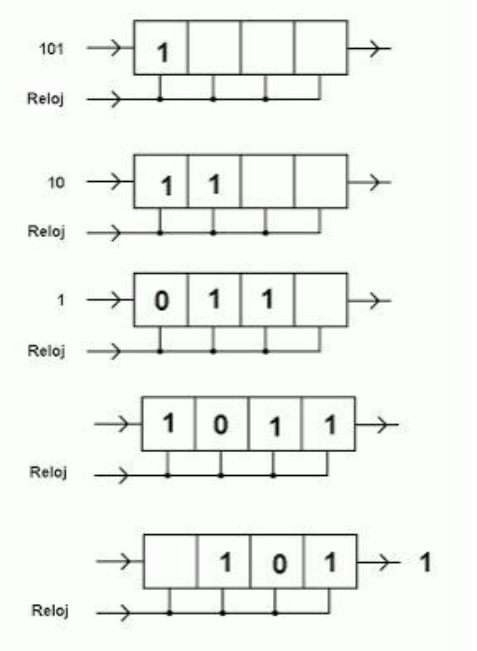
14. Dibuje el esquema de compuertas que componen un flip flop S-R. Describa a través de una tabla los estados en función de las entradas. Modifique el esquema anterior para hacerlo sincrónico. Describa gráficamente su respuesta temporal.

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.

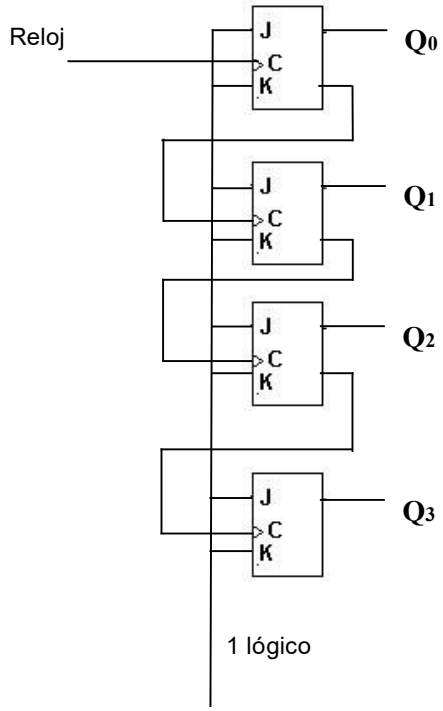
16. Dibuje el esquema de un flip flop J-K, describiendo su respuesta temporal.

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

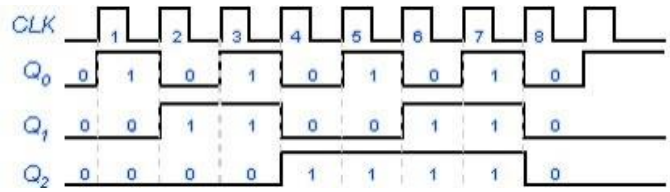
Ayuda: Ejemplo de respuesta temporal para interpretar como responde el registro previo ante la entrada serial del número binario 1011:



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.

Organización de Computadoras 2023

Práctica 4

Assembly, Instrucciones y Simulador MSX88

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)**;

a) Implemente el código utilizando máquinas de 1, 2 y 3 direcciones.
b) Realice una tabla de comparación similar a la del ejercicio 2.
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 = 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
NUM0    DB  0CAh
NUM1    DB  0
NUM2    DW  ?
NUM3    DW  0ABCDh
NUM4    DW  ?
```

```
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
```

Cuestionario:

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

Cuestionario:

1) ¿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?
2) ¿Qué cadenas binarias representan a NUM1 y NUM2 en la memoria del simulador? ¿En qué sistemas binarios están expresados estos valores?
3) 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
INI   DB  0
FIN   DB  15
```

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

Cuestionario:

a) ¿Cuántas veces se ejecuta el lazo? ¿De qué variables depende esto en el caso general?
b) 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:
   1º) JS
   2º) JZ
   3º) 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:

a) A ≤ 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:

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

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
```

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.