UNIVERSIDAD AUTÓNOMA DEL CARMEN

**FACULTAD DE COMERCIO YA ADMINISTRACIÓN**

**LICENCIATURA EN INFORMÁTICA**

Índice

Contenido

[UNIDAD I. 4](#_Toc105493424)

[I.1 Máquina virtual. 4](#_Toc105493425)

[I.2 Generaciones de computadoras. 6](#_Toc105493426)

[Practica 1 6](#_Toc105493427)

[Objetivo: 6](#_Toc105493428)

[Desarrollo de la práctica: 6](#_Toc105493429)

[Entregables: 6](#_Toc105493430)

[I.2.- Representación de números en diferentes bases. 6](#_Toc105493431)

[Practica 2 7](#_Toc105493432)

[Objetivo: 7](#_Toc105493433)

[Desarrollo de la práctica: 7](#_Toc105493434)

[Entregables: 7](#_Toc105493435)

[I.3.- Representación interna de los datos. 7](#_Toc105493436)

[I.4. Aritmética binaria. 10](#_Toc105493437)

[Practica 3 11](#_Toc105493438)

[Objetivo: 11](#_Toc105493439)

[Desarrollo de la práctica: 11](#_Toc105493440)

[Entregables: 11](#_Toc105493441)

[I4.- Estructura básica de la computadora. 11](#_Toc105493442)

[UNIDAD II. El nivel de lógica digital. 13](#_Toc105493443)

[2.1 Álgebra booleana. 13](#_Toc105493444)

[2.2 Postulados del álgebra booleana. 13](#_Toc105493445)

[2.3 Funciones. 14](#_Toc105493446)

[Practica 4 16](#_Toc105493447)

[Objetivo: 16](#_Toc105493448)

[Desarrollo de la práctica: 16](#_Toc105493449)

[Entregables: 16](#_Toc105493450)

[2.4 Mapas de karnaught. 16](#_Toc105493451)

[Practica 5 20](#_Toc105493452)

[2.5 Reducción por el método de Quine Mc. Kluskey. 20](#_Toc105493453)

[Practica 6 24](#_Toc105493454)

[2.6. Compuertas lógicas. 26](#_Toc105493455)

[2.6.1 Implementación de funciones con compuertas lógicas. 29](#_Toc105493456)

[Practica 7 31](#_Toc105493457)

[2.7. Organización de la memoria. 33](#_Toc105493458)

[2.7 dispositivos de entrada salida. 34](#_Toc105493459)

[UNIDAD III EL NIVEL DE MICROPROGRAMACIÓN. 35](#_Toc105493460)

[III.1. TRAYECTORIA DE LOS DATOS. 35](#_Toc105493461)

[III.2. MICROINSTRUCCIONES. 35](#_Toc105493462)

[III.3. MACROINSTRUCCIONES. 35](#_Toc105493463)

[III.4. CRONOLOGÍA. 35](#_Toc105493464)

[III.5. SECUENCIA. 35](#_Toc105493465)

[III.6. PROCESAMIENTO PARALELO. 35](#_Toc105493466)

[UNIDAD IV MÁQUINA CONVENCIONAL (LENGUAJE DE MÁQUINA). 35](#_Toc105493467)

[IV.1. FORMATO DE INSTRUCCIONES. 35](#_Toc105493468)

[IV.2. DIRECCIONAMIENTO. 35](#_Toc105493469)

[IV.3. TIPOS DE INSTRUCCIONES. 35](#_Toc105493470)

[IV.4. FLUJO DE CONTROL. 35](#_Toc105493471)

[UNIDAD V EL SISTEMA OPERATIVO. 35](#_Toc105493472)

[V.1. MEMORIA VIRTUAL. 35](#_Toc105493473)

[V.1.1 PAGINACIÓN 35](#_Toc105493474)

[V.1.2. SEGMENTACIÓN. 35](#_Toc105493475)

[V.2. INSTRUCCIONES VIRTUALES DE E/S. 35](#_Toc105493476)

[V.3. PROCESAMIENTO PARALELO. 35](#_Toc105493477)

[V.4. CASOS DE ESTUDIO. 35](#_Toc105493478)

[V.4.1. MS-DOS. 35](#_Toc105493479)

[V.4.2. UNÍS. 35](#_Toc105493480)

[UNIDAD VI EL LENGUAJE ENSAMBLADOR. 35](#_Toc105493481)

[VI.1. SEGMENTACIÓN. 35](#_Toc105493482)

[VI.2. DIRECCIONAMIENTO. 35](#_Toc105493483)

[VI.3. TIPOS DE INSTRUCCIONES. 35](#_Toc105493484)

[VI.4. FUNCIONES Y PROCEDIMIENTOS. 35](#_Toc105493485)

[VI.5. MACROINSTRUCCIONES. 36](#_Toc105493486)

[VI.6. LENGUAJES DE ALTO NIVEL. 36](#_Toc105493487)

[UNIDAD VII ARQUITECTURAS AVANZADAS. 36](#_Toc105493488)

[VII.1. PIPELINE. 36](#_Toc105493489)

[VII.2. MÁQUINAS RISC. 36](#_Toc105493490)

[VII.3. ARQUITECTURAS EN PARALELO. 36](#_Toc105493491)

# UNIDAD I.

Introducción

## I.1 Máquina virtual.

La computadora funciona como un conjunto de máquinas virtuales anidadas, ¿que es una máquina virtual? Si usted observa su celular por ejemplo ve imágenes, sombras colores, pero eso no existe, como usted sabe una computadora internamente solo almacena y procesa ceros y unos, la computadora mediante software y hardware nos da la impresión de que hay imágenes, y letras, aunque internamente eso no exista pues dentro solo encontramos ceros y unos y eso es una máquina virtual, es una máquina que nos da la percepción de una realidad que simula mediante algoritmos. En una computadora podemos encontrar 6 niveles de máquina virtual una sobre otra que nos va dando una mejor percepción (más avanzada) cada vez que ascendemos.

Lenguaje de Alto Nivel

Nivel de Lenguaje Ensamblador

Nivel del Sistema Operativo

Nivel de Máquina Convencional (“Lenguaje de Máquina”)

Nivel de Microprogramación

Nivel de Lógica Digital

Traducción Compilador

Traducción ensamblador

Interpretación parcial S.O. (Interrupciones).

Interpretación (Microprograma)

Los microprogramas se ejecutan directamente por el Hardware

Nivel 4

Nivel 5

Nivel 3

Nivel 2

Nivel 1

Nivel 0

Como ejemplo, piense que está en el nivel 5, en el de lenguaje de alto nivel por ejemplo C++, su cliente le pide una “máquina virtual” que lleve el control de su almacén (usted le llamaría programa), usted para programar no usa ceros y unos usted ve un equipo que tiene un compilador que recibe órdenes en C++ y que puede comunicarse con el disco la red, la base de datos, usted ve una máquina virtual, y cuando entrega su programa al cliente, el cliente no ve sentencias de código en C++, su cliente ve ventanas para listar productos, dar de alta, baja, modificaciones, su cliente ve una máquina virtual, pero ambos ven una máquina virtual diferente.

En este curso vamos a ver cómo se van construyendo las diferentes capas de máquina virtual para construir un equipo funcional como los que usted usa.

## I.2 Generaciones de computadoras.

Empezamos con un resumen de los eventos importantes en la evolución de las computadoras:

GENERACIÓN 0: Blaise Pascal (1623-1662) construyo una máquina calculadora en 1642 que nunca funciono al 100%.

GENERACIÓN 1: Bulbos (1945-1955)

ENIAC (Eckert / Mauchley) 1996

GENERACIÓN 2: transistores (1955-1965)

El transistor se inventó en los lab. Bell 1948 principal exponente PDP-1

GENERACIÓN 3:Circuitos Integrados (1965-1980)

generación 4: Computadoras Personales y VLSI (1980-199?).

Transistor

Vea el video:

<https://www.youtube.com/watch?v=ljSdrYmdF44>

## Practica 1

## Objetivo:

comprender la evolución de las computadoras en la historia.

## Desarrollo de la práctica:

Haga una línea de tiempo con los principales eventos que influyeron en la evoluciónd e la computadora desde la antigüedad hasta el día de hoy, documéntese en internet

## Entregables:

Archivo Word y archivo pdf con el resultado de su trabajo

## I.2.- Representación de números en diferentes bases.

Para entender una computadora empezamos por las bases matemáticas. Los números son una abstracción de la realidad, son representaciones de magnitudes, estas representaciones son elegidas de alguna forma, esto es: el símbolo “5” no es el número cinco, si no una representación del mismo, pero NO la única, “V”, es otra representación (estas usaban los romanos), pero ¿porque? “5”. Bueno en la numeración arábiga, se eligió la base 10, esto quiere decir que tenemos 10 símbolos para representar magnitudes, a saber: (0,1,2,3,4,5,6,7,8,9) , al igual que se eligió la base 10 se pudo haber elegido la base 6 o la 5 o la 2 o cualquier otra, esto en un sentido práctico repercute esencialmente en la cantidad de símbolos para la representación de magnitudes, supongamos la base 2, necesitamos un par de símbolos, estos podrían ser (&,%) o cualesquiera otros, pero si pueden ser cualesquiera, porque no usar un par de símbolos a los que estemos acostumbrados, por ejemplo (0,1), podríamos usar también la base 8 para la que requeriríamos 8 símbolos por ejemplo(0,1,2,3,4,5,6,7) o la base 16 para la que requeriríamos 16 símbolos (0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f) vea que en base 16 como se me acabaron los números use las primeras letras del alfabeto para completar los 16 símbolos. En seguida se muestra una tabla de magnitudes en diferentes bases:

|  |  |  |  |
| --- | --- | --- | --- |
| Base 10 | Base 2 | Base 8 | base 16 |
| 00 | 00000 | 00 | 00 |
| 01 | 00001 | 01 | 01 |
| 02 | 00010 | 02 | 02 |
| 03 | 00011 | 03 | 03 |
| 04 | 00100 | 04 | 04 |
| 05 | 00101 | 05 | 05 |
| 06 | 00110 | 06 | 06 |
| 07 | 00111 | 07 | 07 |
| 08 | 01000 | 10 | 08 |
| 09 | 01001 | 11 | 09 |
| 10 | 01010 | 12 | 0a |
| 11 | 01011 | 13 | 0b |
| 12 | 01100 | 14 | 0c |
| 13 | 01101 | 15 | 0d |
| 14 | 01110 | 16 | 0e |
| 15 | 01111 | 17 | 0f |
| 16 | 10000 | 20 | 10 |
| 17 | 10001 | 21 | 11 |
| 18 | 10010 | 22 | 12 |

## Practica 2

## Objetivo:

Manejar las diferentes bases numéricas.

## Desarrollo de la práctica:

Realice los siguientes ejercicios:

1. Escriba un algoritmo que permita contar de uno en uno en base 10.
2. Escriba un algoritmo que describa como contar en base 3
3. Generalice para cualquier base.

## Entregables:

Archivo Word y archivo pdf con el resultado de su trabajo

## I.3.- Representación interna de los datos.

Al interior de la computadora se manejan solo dos estados(símbolos), voltaje y no voltaje, por lo que con esto solo podemos utilizar la base 2, todo número entero será representado en base 2, (voltaje=1, no voltaje=0), los números reales son representados en notación científica y los caracteres mediante el código ASCII, en fin que a pesar de que la computadora solo reconoce dos estados, no es impedimento para que pueda representar los datos tal y como los entiende la mente humana.

Para representar sus datos, la computadora usa posiciones (pensemos en terminales eléctricas) a las que llamamos bits y en las que puede representar un 0 o un 1. Recordando el principio fundamental del conteo, suponga que tiene un conjunto A con 3 elementos y un conjunto B con 2 elementos, ¿de cuantas formas se puede tomar un elemento del conjunto A y uno de B?, la respuesta es cordialidad(A)\*cordialidad(B), ahora si no fueran conjuntos diferentes, ¿de cuantas formas podemos tomar 2 elementos de A con repetición?, la respuesta es 3\*3= 3^2. Esta es la razón por la que las capacidades de almacenamiento de las computadoras se miden en potencias de 2 pues disponemos de un conjunto de 2 elementos y en dependencia del número de bits es la capacidad de almacenaje de una magnitud (y con esto de un direccionamiento como veremos mas adelante).

1.3 Cambio de base.

Para cambiar de base 10 a una base X, tome el número en base 10 y realice divisiones enteras sucesivas, para la primera división tome como dividendo el número en base 10, para la división (n+1) tome como dividendo el cociente de la división (n) y vaya almacenando los residuos. Para todas las divisiones tome como divisor (X), es decir la base a la que se quiere ir. finalmente coloque los residuos en orden inverso y esta es la representación en la base (X).

Para cambiar de una base X a base 10, suponga que xi es el dígito que ocupa la posición i de derecha a izquierda en la cifra (ejemplo si B=01101, x0=1, x1=0, x2=1, etc.) y calcule la sumatoria: Σ(xi\*(X^i)), donde X es la base origen. Por ejemplo, para convertir el binario B=01101 a decimal Σ(xi\*(2^i)) = (0\*2^4+1\*2^3+1\*2^2+0\*2^1+1\*2^0) = 13.

Para convertir base 16 a base 2, tome cada digito de la cifra en base 16 y sustitúyalo por cuatro dígitos binarios mediante una tabla, rellenando con ceros a la izquierda de ser necesario. Para convertir de base 2 a base 16, ordene paquetes de 4 dígitos de la cifra binaria de derecha a izquierda y sustituya por su correspondiente dígito hexadecimal.

En el caso de base 8, proceda como en la base 16 pero con 3 dígitos.

Ahora los números se representan en binario en BCD (decimal codificado en binario) como se muestra en la tabla:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | decimal |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | 10 |
| 1 | 0 | 1 | 1 | 11 |
| 1 | 1 | 0 | 0 | 12 |
| 1 | 1 | 0 | 1 | 14 |
| 1 | 1 | 1 | 0 | 14 |
| 1 | 1 | 1 | 1 | 15 |

¿Pero cómo representar los negativos?

**Representación de números negativos.**

* Complemento a unos.
* Signo y magnitud.
* Complemento a 2.

Signo y magnitud. Esta representación toma el primer bit de la representación binaria y con este bit se representa el signo del número, si es 0 es positivo y si es 1 es negativo:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | magnitud | Signo y magnitud |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 | 2 |
| 0 | 0 | 1 | 1 | 3 | 3 |
| 0 | 1 | 0 | 0 | 4 | 4 |
| 0 | 1 | 0 | 1 | 5 | 5 |
| 0 | 1 | 1 | 0 | 6 | 6 |
| 0 | 1 | 1 | 1 | 7 | 7 |
| 1 | 0 | 0 | 0 | 8 | -0 |
| 1 | 0 | 0 | 1 | 9 | -1 |
| 1 | 0 | 1 | 0 | 10 | -2 |
| 1 | 0 | 1 | 1 | 11 | -3 |
| 1 | 1 | 0 | 0 | 12 | -4 |
| 1 | 1 | 0 | 1 | 14 | -5 |
| 1 | 1 | 1 | 0 | 14 | -6 |
| 1 | 1 | 1 | 1 | 15 | -7 |

Esta es la representación solo tiene un problema que tiene 2 veces el 0 un cero positivo y un cero negativo.

Otra representación es el complemento a 1, se toma cada número y para obtener su negativo intercámbianos todos los bits, si es 0 se convierte en 1 y si es 1 se convierte en 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | magnitud | Complemento a 1 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 | 2 |
| 0 | 0 | 1 | 1 | 3 | 3 |
| 0 | 1 | 0 | 0 | 4 | 4 |
| 0 | 1 | 0 | 1 | 5 | 5 |
| 0 | 1 | 1 | 0 | 6 | 6 |
| 0 | 1 | 1 | 1 | 7 | 7 |
| 1 | 0 | 0 | 0 | 8 | -7 |
| 1 | 0 | 0 | 1 | 9 | -6 |
| 1 | 0 | 1 | 0 | 10 | -5 |
| 1 | 0 | 1 | 1 | 11 | -4 |
| 1 | 1 | 0 | 0 | 12 | -3 |
| 1 | 1 | 0 | 1 | 14 | -2 |
| 1 | 1 | 1 | 0 | 14 | -1 |
| 1 | 1 | 1 | 1 | 15 | -0 |

Esta forma de representar negativos tiene la ventaja de que es fácil pasar de positivo a negativo simplemente aplicando el complemento, pero sigue teniendo 2 ceros.

## I.4. Aritmética binaria.

De la misma forma que existe la álgebra en base 10 con sus axiomas y teoremas, existe una álgebra en base 2, un matemático del siglo XIX, trabajo mucho sobre esto, razón por la que en honor a su nombre se le llama también Álgebra Booleana, los axiomas y teoremas de esta álgebra, no son exactamente igual a los del álgebra de base 10, pero algunas cosas se siguen cumpliendo, por ejemplo, se tienen las operaciones suma y producto que son también cerradas en el conjunto de los números binarios, en el siguiente apartado se verá algo de esta álgebra, por ahora solo veremos algunas operaciones aritméticas en binario.

Operaciones aritméticas.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | magnitud | complemento a 2 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 | 2 |
| 0 | 0 | 1 | 1 | 3 | 3 |
| 0 | 1 | 0 | 0 | 4 | 4 |
| 0 | 1 | 0 | 1 | 5 | 5 |
| 0 | 1 | 1 | 0 | 6 | 6 |
| 0 | 1 | 1 | 1 | 7 | 7 |
| 1 | 0 | 0 | 0 | 8 | -8 |
| 1 | 0 | 0 | 1 | 9 | -7 |
| 1 | 0 | 1 | 0 | 10 | -6 |
| 1 | 0 | 1 | 1 | 11 | -5 |
| 1 | 1 | 0 | 0 | 12 | -4 |
| 1 | 1 | 0 | 1 | 14 | -3 |
| 1 | 1 | 1 | 0 | 14 | -2 |
| 1 | 1 | 1 | 1 | 15 | -1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 0 | 1 | 1 | 1 |
|  | 1 | 0 | 0 | 0 |
| + |  |  |  | 1 |
| -7 | 1 | 0 | 0 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -7 | 1 | 0 | 0 | 1 |
| +2 | 0 | 0 | 1 | 0 |
|  | 1 | 0 | 1 | 1 |

## Practica 3

## Objetivo:

Manejar las diferentes bases numéricas.

## Desarrollo de la práctica:

Realice los siguientes ejercicios:

1. Si una computadora representa sus enteros con 6 bits, cuál sería el menor negativo y el mayor positivo que esta podría manejar.
2. convierta el valor ABCD en hexadecimal a binario.
3. escriba un algoritmo en lenguaje natural para la conversión de base 2 a base 4.
4. realice las siguientes operaciones a 8 bits en complemento a 2. 255-64, 143+79, 45-45.

## Entregables:

Archivo Word y archivo pdf con el resultado de su trabajo

## I4.- Estructura básica de la computadora.

Máquina de Von Neumann.

Memoria

Unidad

de

Control

Unidad

Aritmético Acc

Lógica

Entrada

Salida

ACC= acumulador

# UNIDAD II. El nivel de lógica digital.

## 2.1 Álgebra booleana.

El álgebra de Boole inicialmente se definió para trabajar con la lógica proposicional donde se trabaja con proposiciones que pueden ser falsas o verdaderas y esto se corresponde al binario si tomamos 1 como cierto y 0 como falso. De la misma forma que existe un algebra para base 10 que es la que conocemos y en la que se dan los postulados que la sustentan (vea: “<http://www.uamenlinea.uam.mx/materiales/matematicas/alg_basica/ADALID_DIEZ_DE_U_CLARAMARTHA_Fundamentos_de_algebra.pdf>” los axiomas de los números página 18 a 20), también existe un algebra para base 2 con un conjunto de postulados.

El álgebra booleana opera sobre el conjunto {0,1} sobre el que se definen las operaciones \*, + y ~ también llamadas conjunción, disyunción y negación o and y or y not.

|  |  |  |  |
| --- | --- | --- | --- |
| Tabla Operadores del algebra booleana | | | |
| \* | y | and | Conjunción |
| + | o | or | Disyunción |

14Tabla de operadores del algebra booleana

|  |  |  |  |
| --- | --- | --- | --- |
| Tabla de operadores | | | |
| a | b | a\*b | a+b |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

15Tabla de verdad de operadores del algebra booleana

**Nota. Note que en el and (\*) el resultado es 0 si alguna de las entradas es 0, y en el or (+) el resultado es 1 si alguna de las entradas es 1.**

|  |  |
| --- | --- |
| Tabla del not | |
|  | |
| a | a' |
| 0 | 1 |
| 1 | 0 |

## 2.2 Postulados del álgebra booleana.

|  |  |  |
| --- | --- | --- |
| Cerradura sobre el operador + | | |
| Cerradura sobre el operador \* | | |
| X+0=X. | X\*1=X | Neutro |
| X\*0=0 | X+1=1 | Neutro |
| X+X’=1 | X\*X’=0 |  |
| X+Y=Y+X | X\*Y=Y\*X | Asociativa |
| X+(Y+Z)=(X+Y)+Z | X\*(Y\*Z)=(X\*Y)\*Z | Distributiva |
| X\*(Y+Z)=X\*Y+X\*Z | X+(Y\*Z)=(X+Y)\*(X+Z) | Distributiva |
| (X+Y)=X’\*Y’ | (X\*Y)=X’+Y’ | DEMORGAN |
| A(A+B)=A | A+(A\*B)=A | ABSORCIÓN |
| (X’)’=X |  | IDEMPOTENCIA |

+ equivale al or matemático

\* equivale al and matemático

~,’ o raya sobre la variable equivale a la negación matemática p. Ej. La negación de X se representa como:

\_

X o como ~X o como X’.

**Nota: Al igual que en la base 10 en la operación (X\*Y) se acostumbra omitir el \* y poner (XY).**

## 2.3 Funciones.

Una función f (A, B) es una regla que asocia un dígito binario para una combinación de valores de A y B con A, B variables binarias.

Una función se puede describir por extensión o por comprensión, por extensión se listan todos sus valores y por comprensión se define una regla, por ejemplo en base 10 la función f(x)=x^2, es una regla que indica que se debe tomar el valor x multiplicarlo por sí mismo y ese es el valor de la función, así f(2)=4, f(3)=9 etc... Esta es una función descrita por comprensión, ahora suponga la función f(x)=x+1 con x un elemento del conjunto {1,2,3,4}, la función esta descrita por comprensión, pero como su dominio es tan pequeño, se puede describir también por extensión y queda:

|  |  |
| --- | --- |
| x | f(x) |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |

En binario como los valores de la función son finitos, se acostumbra describir las funciones por extensión.

Ejemplos:

|  |  |  |
| --- | --- | --- |
| a | b | F(a,b) |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | F(A,B,C) |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Miniterminos**

Otra forma de representar las funciones es la siguiente:

Cada renglón donde la función se hace 1 se llama mini termino por lo que otra forma de representar la función anterior es: f(A,B,C)= m2,m4,m6 pues en la tabla la función se hace 1 en los renglones 2, 4 y 6 si contamos desde 0.

**Maxiterminos**

Otra forma de representar las funciones es la siguiente:

Cada renglón donde la función se hace 0 se llama maxi termino por lo que otra forma de representar la función anterior es: f(A,B,C)= M0,M1,M3,M5,M7 pues en la tabla la función se hace 0 en los renglones 0, 1, 3, 5 y 7 si contamos desde 0.

**Suma de mini términos.**

Una función booleana se puede expresar usando las compuertas lógicas and, or y not usando la siguiente regla:

1. se toman todos los renglones de la función donde esta se hace uno.
2. Para cada renglón se revisan los valores de las variables de la función, si el valor es 0 se pone la variable negada, si tiene un uno se pone sin negar, separando las variables por una compuerta and.
3. Se agrega una compuerta or para separar cada repetición del punto 2.

Ejemplo: para la función:

|  |  |  |
| --- | --- | --- |
| a | b | F(a,b) |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

tenemos:

F(a,b)=~a~b+~ab+a~b

**Producto de maxi términos.**

Una función booleana se puede expresar usando las compuertas lógicas and, or y not usando la siguiente regla:

1. se toman todos los renglones de la función donde esta se hace cero.
2. Para cada renglón se revisan los valores de las variables de la función, si el valor es 1 se pone la variable negada, si tiene un cero se pone sin negar, separando las variables por una compuerta or.
3. Se agrega una compuerta and para separar cada repetición del punto 2.

Para la tabla anterior tenemos:

F(a,b)= (~a+b)

## Practica 4

## Objetivo:

Implementar funciones binarias.

## Desarrollo de la práctica:

Realice los siguientes ejercicios:

1. Muestre las tablas de las siguientes funciones:
   1. f(a,b,c)=m0,m1,m4,m5,m7
   2. f(a,b,c,d)=M0,M3,M7,M9,M11,M12
   3. f(a,b,c,d)=m1,m3,m4,m7,m8
   4. F(a,b,c)=m2+m3+m4+m5++m6+m7
   5. F(a,b,c)=m0+m1+m3+m5
2. Exprese las funciones del ejercicio 1 como:
   1. Suma de miniterminos
   2. Producto de maxiterminos

## Entregables:

Archivo Word y archivo pdf con el resultado de su trabajo

## 2.4 Mapas de karnaught.

El mapa de Karnaugh o mapa-k es un diagrama utilizado para la simplificación de funciones algebraicas Booleanas, permitiendo de manera gráfica reconocer patrones y así reduce la necesidad de hacer cálculos extensos para la simplificación de expresiones booleanas. Tiene las siguientes ventajas:

* El mapa-k nos permite convertir la tabla de verdad de una ecuación booleana en una forma SOP(Suma de productos) o POS(Productos de suma) minimizada.
* Reglas básicas y sencillas para la simplificación.
* La facilidad del método permite que sea más rápido y más eficiente que otras técnicas de simplificación en el Álgebra de Boole.

Reglas para crear MK

Mapa de 3 variables:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a\bc | 00 | 01 | 11 | 10 |
| 0 |  |  |  |  |
| 1 |  |  |  |  |

Mapa de 4 variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab\cd | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  |  |  |
| 11 |  |  |  |  |
| 10 |  |  |  |  |

Mapa de 5 variables

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a=0 |  |  |  |  |  | a=1 |  |  |  |  |
| bc\de | 00 | 01 | 11 | 10 |  | bc\de | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |  | 00 |  |  |  |  |
| 01 |  |  |  |  |  | 01 |  |  |  |  |
| 11 |  |  |  |  |  | 11 |  |  |  |  |
| 10 |  |  |  |  |  | 10 |  |  |  |  |

Observe que las columnas 11 y 10 están intercambiadas en los mapas.

**Llenar los mapas**

Simplemente se hace corresponder las combinaciones de variables con los valores de la función.

Ejemplo 1

F(a,b,c)=

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | c | F |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Llenar el mapa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a\bc | 00 | 01 | 11 | 10 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Para reducir este mapa se deben hacer grupos donde:

* Se hacen rectángulos que encierran cada grupo
* Cada grupo solo contiene unos
* la cantidad de unos debe ser potencias de 2. Podemos hacer grupos de 1,2,4,8, 16 unos, y cada grupo debe contener solo unos.
* Todos los unos deben quedar en al menos un grupo.
* Se debe hacer la menor cantidad de grupos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a/bc | 00 | 01 | 11 | 10 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Aquí se hicieron 3 grupos

Ahora se analiza cada grupo

1. Se mira cada grupo y en cada grupo se eliminan variables, el número de variables eliminadas en la potencia de 2 correspondiente a la cantidad de unos en el grupo, por ejemplo, para un grupo de 2 unos como el 2 salió de elevar 2^1, se elimina 1 variable, para 4 unos como 4 sale de elevar 2^2 se eliminan 2 variables y para 8 como 8 sale de elevar 2 ^3 se eliminan 3 variables.
2. Ahora para saber que variable se elimina, se analiza cada variable de izquierda a derecha y de arriba abajo del grupo, por ejemplo, el grupo rojo, no tiene arriba y abajo pues es solo un renglón, pero tiene izquierda y derecha pues abarca dos columnas, de izquierda a derecha en el grupo rojo observamos que la variable c cambia de 0 a 1 por tanto se elimina, la variable a y la b no cambian y el resultado del circulo es el producto de las variables que no se eliminan poniéndolas negadas si están en 0 , en el círculo rojo= ~a\*~b.
3. Se realiza la misma operación con todos los curculos y al final se suman los resultados

F(a,c,c)= ~a\*~b+b\*c+ab

Ejemplo 2:

Para un mapa como este:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a/bc | 00 | 01 | 11 | 10 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |

Imagínese el mapa como una cartulina que dobla horizontalmente y pega por atrás, este mapa visto por atrás queda:

|  |  |  |
| --- | --- | --- |
| a/bc | 00 | 10 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

Y podemos circular 4 unos.

Ejemplo 3:

Suponga el siguiente mapa

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a=0 |  |  |  |  |  | a=1 |  |  |  |  |
| bc/de | 00 | 01 | 11 | 10 |  | bc/de | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 0 | 1 |  | 00 | 1 | 0 | 0 | 1 |
| 01 | 0 | 0 | 0 | 0 |  | 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 0 | 0 | 0 |  | 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 1 |  | 10 | 1 | 0 | 0 | 1 |

Cuando tiene este tipo de mapa, para hacer grupos de unos:

primero debe recordar que cada grupo debe ser lo más grande posible.

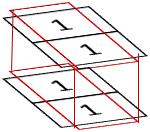
Imagínese el mapa en 3 dimensiones de manera que la parte a=0 está encima de la parte a=1.

En lugar de rectángulos, busque hacer cubos tomando los unos de ambos grupos

En este ejemplo puede hacer grupos de

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a=0 |  |  |  |  |  | a=1 |  |  |  |  |
| bc\de | 00 | 01 | 11 | 10 |  | bc\de | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |  | 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |  | 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 0 | 0 | 0 |  | 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 |  | 10 | 1 | 0 | 0 | 0 |

De tal manera que se vería así:



Estudie el punto 3.3 de la liga:

<https://bookdown.org/alberto_brunete/intro_automatica/mapa-de-karnaugh.html>

## Practica 5

Objetivo

Simplificar funciones boolenas por el método de mapas de karnaugth

Desarrollo

Simplifique las siguientes funciones:

1. F(a,b,c)=m2+m3+m4+m5++m6+m7
2. F(a,b,c)=m0+m1+m3+m5

Entregables

## 2.5 Reducción por el método de Quine Mc. Kluskey.

También llamado método tabular, se utiliza para reducir ecuaciones booleanas. El método se divide en dos partes: encontrar los implicantes primos y obtener las ecuaciones a partir de la tabla de implicantes primos.

Encontrar implicantes primos.

1. Se toman los mintérminos de la tabla de verdad, y se convierten a su equivalente en binario. Σm(0,1,2,4,5,7,8,9,10,12,13, 15)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | **Z** |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

1. Se colocan en la Columna I, los mintérminos ordenados de menor a mayor número de unos.

Columna I

0 0 0 0 0

O unos

1 0 0 0 1

2 0 0 1 0

4 0 1 0 0

4 unos

3 unos

2 unos

1 unos

|  |  |
| --- | --- |
| 8 | 1 0 0 0 |
| 5 | 0 1 0 1 |
| 9 | 1 0 0 1 |
| 10 | 1 0 1 0 |
| 12 | 1 1 0 0 |
| 7 | 0 1 1 1 |
| 13 | 1 1 0 1 |
| 15 | 1 1 1 1 |

1. Se toma cada elemento de cada grupo y se comparan los renglones del siguiente grupo que sólo tienen una diferencia en sus bits, formando la siguiente columna. En esta columna se escriben los mintérminos comparados y y el nuevo término, donde se marcará con un guion ( \_ ) esa diferencia. Cada término que pase a la siguiente columna deberá marcarse ( )

Columna I Columna II

0 0 0 0 0  (0,1) 0 0 0 \_

1 0 0 0 1 

2 0 0 1 0

4 0 1 0 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 1 0 0 0 |  | | | | | |
| 5 | 0 1 0 1 |
| 9 | 1 0 0 1 |
| 10 | 1 0 1 0 |
| 12 | 1 1 0 0 |
| 7 | 0 1 1 1 |
| 13 | 1 1 0 1 |
| 15 | 1 1 1 1 |
|  | **Columna I** | **Columna II** | | | | | |
| 0 | 0 0 0 0 |  | (0,1) | 0 | 0 | 0 | \_ |
| 1 | 0 0 0 1 |  | (0,2) | 0 | 0 | \_ | 0 |
| 2 | 0 0 1 0 |  | (0,4) | 0 | \_ | 0 | 0 |
| 4 | 0 1 0 0 |  | (0,8) | \_ | 0 | 0 | 0 |
| 8 | 1 0 0 0 |  | (1,5) | 0 | \_ | 0 | 1 |
| 5 | 0 1 0 1 |  | (1,9) | \_ | 0 | 0 | 1 |
| 9 | 1 0 0 1 |  | (2,10) | \_ | 0 | 1 | 0 |
| 10 | 1 0 1 0 |  | (4,5) | 0 | 1 | 0 | \_ |
| 12 | 1 1 0 0 |  | (4,12) | \_ | 1 | 0 | 0 |
| 7 | 0 1 1 1 |  | (8,9) | 1 | 0 | 0 | \_ |
| 13 | 1 1 0 1 |  | (8,10) | 1 | 0 | \_ | 0 |
| 15 | 1 1 1 1 |  | (8,12) | 1 | \_ | 0 | 0 |
|  |  |  | (5,7) | 0 | 1 | \_ | 1 |
|  |  |  | (5,13) | \_ | 1 | 0 | 1 |
|  |  |  | (9,13) | 1 | \_ | 0 | 1 |
|  |  |  | (12,13) | 1 | 1 | 0 | \_ |
|  |  |  | (7,15) | \_ | 1 | 1 | 1 |
|  |  |  | (13,15) | 1 | 1 | \_ | 1 |

1. El paso 3 se repetirá hasta que ya no sea posible formar nuevas columnas, con eso obtenemos la columna 2, se repite el paso 2 y 3 para formar la columna 3, se vuelve a ordenar y combinar para formar la columna 4.

Columna I Columna II Columna III

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 0 0 0 |  | (0,1) | 0 | 0 | 0 | \_ |  | (0,1,4,5) | 0 | \_ | 0 | \_ |
| 1 | 0 0 0 1 |  | (0,2) | 0 | 0 | \_ | 0 |  | (0,1,8,9) | \_ | 0 | 0 | \_ |
| 2 | 0 0 1 0 |  | (0,4) | 0 | \_ | 0 | 0 |  | (0,2,8,10) | \_ | 0 | \_ | 0 |
| 4 | 0 1 0 0 |  | (0,8) | \_ | 0 | 0 | 0 |  | (0,4,1,5) | 0 | \_ | 0 | \_ |
| 8 | 1 0 0 0 |  | (1,5) | 0 | \_ | 0 | 1 |  | (0,4,8,12) | \_ | \_ | 0 | 0 |
| 5 | 0 1 0 1 |  | (1,9) | \_ | 0 | 0 | 1 |  | (0,8,1,9) | \_ | 0 | 0 | \_ |
| 9 | 1 0 0 1 |  | (2,10) | \_ | 0 | 1 | 0 |  | (0,8,2,10) | \_ | 0 | \_ | 0 |
| 10 | 1 0 1 0 |  | (4,5) | 0 | 1 | 0 | \_ |  | (0,8,4,12) | \_ | \_ | 0 | 0 |
| 12 | 1 1 0 0 |  | (4,12) | \_ | 1 | 0 | 0 |  | (1,5,9,13) | \_ | \_ | 0 | 1 |
| 7 | 0 1 1 1 |  | (8,9) | 1 | 0 | 0 | \_ |  | (1,9,5,13) | \_ | \_ | 0 | 1 |
| 13 | 1 1 0 1 |  | (8,10) | 1 | 0 | \_ | 0 |  | (4,5,12,13) | \_ | 1 | 0 | \_ |
| 15 | 1 1 1 1 |  | (8,12) | 1 | \_ | 0 | 0 |  | (4,12,5,13) | \_ | 1 | 0 | \_ |
|  |  |  | (5,7) | 0 | 1 | \_ | 1 |  | (8,9,12,13) | 1 | \_ | 0 | \_ |
|  |  |  | (5,13) | \_ | 1 | 0 | 1 |  | (8,12,9,13) | 1 | \_ | 0 | \_ |
|  |  |  | (9,13) | 1 | \_ | 0 | 1 |  | (5,7,13,15) | \_ | 1 | \_ | 1 |
|  |  |  | (12,13) | 1 | 1 | 0 | \_ |  | (5,13,7,15) | \_ | 1 | \_ | 1 |
|  |  |  | (7,15) | \_ | 1 | 1 | 1 |  |  |  |  |  |  |
|  |  |  | (13,15) | 1 | 1 | \_ | 1 |  |  |  |  |  |  |

1. Si en alguna de las columnas se repiten elementos, se toma solamente uno para formar la siguiente columna.

Columna III Columna IV

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (0,1,4,5) | 0 | \_ | 0 | \_ |  | (0,1,4,5,8,9,12,13) | \_ | \_ | 0 | \_ |
| (0,1,8,9) | \_ | 0 | 0 | \_ |  | ~~(0,1,8,9,4,5,12,13)~~ | \_ | \_ | 0 | \_ |
| (0,2,8,10) | \_ | 0 | \_ | 0 |  | ~~(0,4,8,12,1,5,9,13)~~ | \_ | \_ | 0 | \_ |
| ~~(0,4,1,5)~~ | 0 | \_ | 0 | \_ |  |  | | | | |
| (0,4,8,12) | \_ | \_ | 0 | 0 |  |
| ~~(0,8,1,9)~~ | \_ | 0 | 0 | \_ |  |
| ~~(0,8,2,10)~~ | \_ | 0 | \_ | 0 |  |
| ~~(0,8,4,12)~~ | \_ | \_ | 0 | 0 |  |
| (1,5,9,13) | \_ | \_ | 0 | 1 |  |
| ~~(1,9,5,13)~~ | \_ | \_ | 0 | 1 |  |
| (4,5,12,13) | \_ | 1 | 0 | \_ |  |
| ~~(4,12,5,13)~~ | \_ | 1 | 0 | \_ |  |
| (8,9,12,13) | 1 | \_ | 0 | \_ |  |
| ~~(8,12,9,13)~~ | 1 | \_ | 0 | \_ |  |
| (5,7,13,15) | \_ | 1 | \_ | 1 |  |
| ~~(5,13,7,15)~~ | \_ | 1 | \_ | 1 |  |

**Tabla de Implicantes primos**

1. Se dibuja una tabla, en las columnas se acomodan los mintérminos.
2. Acomodar en los renglones los términos de la última columna y de las columnas anteriores que no fueron marcados.
3. Se coloca una X en donde cruzan los términos con los mintérminos.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 4 | 5 | 7 | 8 | 9 | 10 | 12 | 13 | 15 |
| (0,1,4,5,8,9,12,13) | X | X |  | X | X |  | X | X |  | X | X |  |
| (0,2,8,10) | X |  | X |  |  |  | X |  | X |  |  |  |
| (5,7,13,15) |  |  |  |  | X | X |  |  |  |  | X | X |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Se agrupan verticalmente las X
2. Las X que quedan solas son las que marcan cuál término pasará a ser parte de la ecuación final. Esta X eliminará a las que se encuentran en su mismo renglón y se deben marcar los mintérminos involucrados.

x x x x x x x x x x x x

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 4 | 5 | 7 | 8 | 9 | 10 | 12 | 13 | 15 |
| (0,1,4,5,8,9,12,13) | X | X |  | X | X |  | X | X |  | X | X |  |
| (0,2,8,10) | X |  | X |  |  |  | X |  | X |  |  |  |
| (5,7,13,15) |  |  |  |  | X | X |  |  |  |  | X | X |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

\*

\*

\*

1. Si al final quedan mintérminos sin marcar, se tomará un término que los involucre, tomando el mismo criterio que en mapas de Karnaugh: agrupar el mayor número de mintérminos en el menor número de grupos posibles.
2. Los guiones representan a las variables que se eliminan, los 1 a las variables y los 0 a las variables negadas, formando cada una de las partes de la ecuación final.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **Ecuación** |
| (0,1,4,5,8,9,12,13) | \_ | \_ | 0 | \_ | C’ |
| (0,2,8,10) | \_ | 0 | \_ | 0 | B’ D’ |
| (5,7,13,15) | \_ | 1 | \_ | 1 | B D |

**Ecuación final:** C’ + B’ D’ + BD

Vea el video: <https://www.youtube.com/watch?v=il_tk_S-OMk>

## Practica 6

Objetivo

Simplificar funciones boolenas por el método de Kine M.

Desarrollo

Simplifique las siguientes funciones:

1. F(ab,c,d,e)= ∑m(0,2,3,5,7,8,10,11,13,15,22,29,30)

Entregables

Archivo pdf con el resultado de su actividad

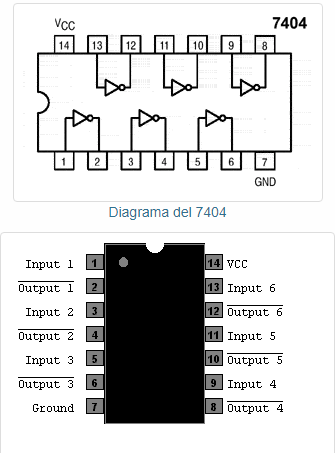
## 2.6. Compuertas lógicas.

Las compuertas lógicas son circuitos electrónicos diseñados para obtener resultados booleanos (0,1), los cuales se obtienen de operaciones lógicas binarias (suma, multiplicación binaria). Dichas compuertas son AND, OR, NOT, NAND, NOR, XOR, XNOR. Además, se pueden conectar entre sí para obtener nuevas funciones. A continuación, se describirá las características de las compuertas. Este tipo de dispositivos lógicos se encuentran implementados con transistores y diodos en un semiconductor y actualmente podemos encontrarlas en formas de circuitos integrados lógicos.

Estos circuitos los puede obtener en integrados:

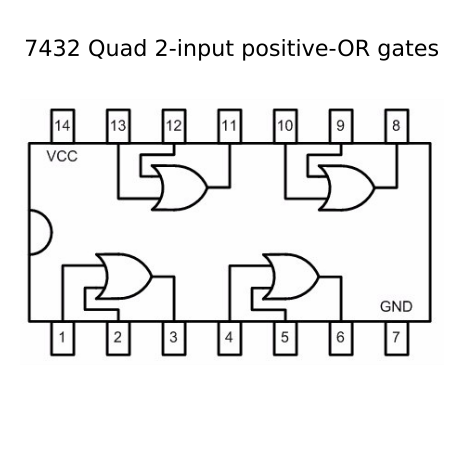


Integrado para la compuerta not (7404) con 14 patas, su diagrama es:

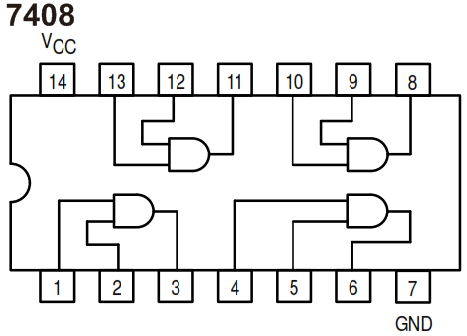


La línea 14 se conecta a 5v y la línea 7 a tierra.

A continuación, el diagrama del 732 (compuerta or):



Y al del 7408 (compuerta and)



Estas compuertas se representan con los siguientes símbolos:

AND, OR, NOT

|  |  |  |
| --- | --- | --- |
| A | B | AND |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

A

B

R

A

B

-

Vcc

A

B

R

|  |  |  |
| --- | --- | --- |
| A | B | OR |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

A

B

R

‗

Vcc

A

R

A

|  |  |
| --- | --- |
| A | Ā |
| 0 | 1 |
| 1 | 0 |

Ā

+

‗‾

Vcc

### 2.6.1 Implementación de funciones con compuertas lógicas.

Para implementar funciones con compuertas proceda así:

Por ejemplo, la función:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **f** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Obtenemos los miniterminos de la función F(a,b,c)=a’bc’+a’bc+ab’c’+abc’+abc

Reduciendo por mapas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a/bc** | **00** | **01** | **11** | **10** |
| **0** | 0 | 0 | 1 | 1 |
| **1** | 1 | 0 | 1 | 1 |

Nos queda:

F(a,b,c)=b+ac’

Implementando con compuertas:

b

c

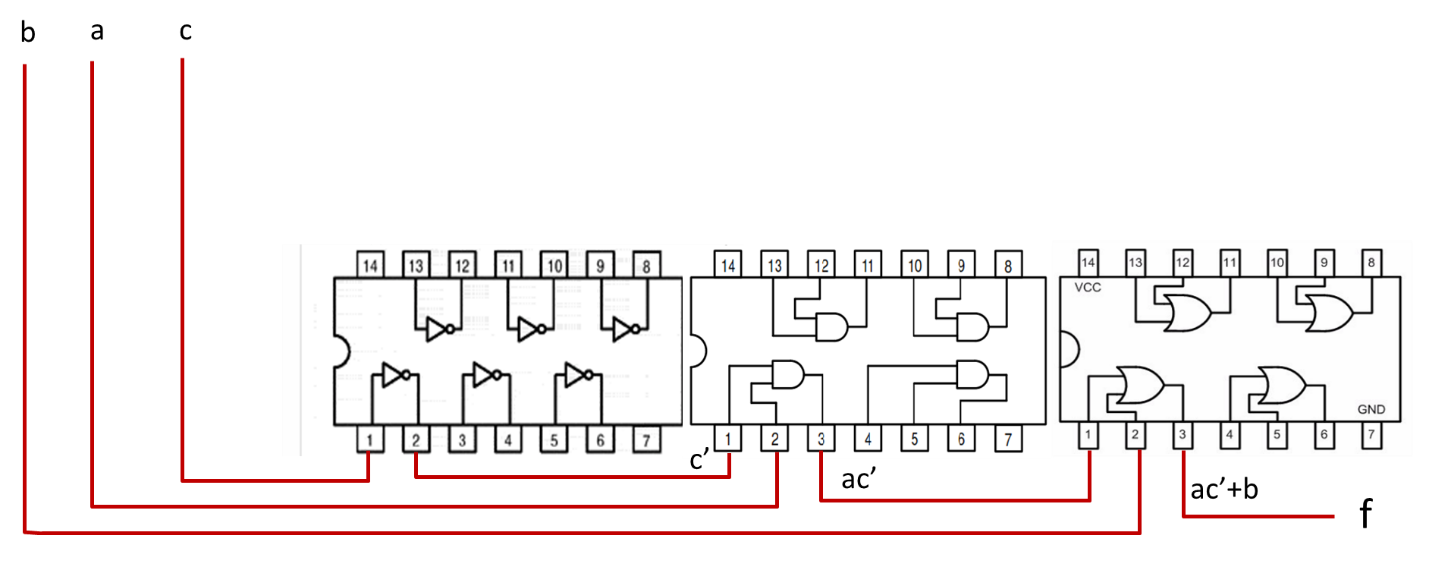
a

c’

ac’

ac’+b

En integrados:



## Practica 7

Objetivo

Implementar funciones lógicas.

Desarrollo

Implemente las funciones:

1)

|  |  |  |
| --- | --- | --- |
| a | b | F(a,b) |
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

2) Implemente la función:

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | c | F(a,b,c) |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Entregables

Archivo pdf con el resultado de su actividad

## Practica 8

Objetivo

Implementar funciones lógicas en el simulador

Desarrollo

Descargue el simulador: ConstructorVirtualDeCircuitos.exe

Implemente las funciones de la práctica 7:

Entregables

Video que muestre los circuitos funcionando.

## 2.7. Organización de la memoria.

Necesitamos un circuito que permita almacenar un dato y conservarlo, empecemos por almacenar y conservar 1 bot, esto sería una memoria de 1 bit, para ello analicemos el flip flop sert reset (SR). Un FF sr es un dispositivo que tiene este diseño:

S

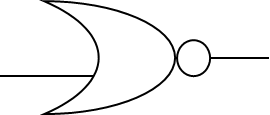
|  |  |  |  |
| --- | --- | --- | --- |
| S | R | Q | Qn+1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | - |
| 1 | 1 | 1 | - |

~Q

Q

R

La compuerta:



Es una compuerta NOR que es un or negado, un or con la salida negada, puede implementarse así:

Esto es una memoria de 1 bit (vea presentación flop flop)

Un fllip flop Tipo D

S

~Q

R

Q

D

S

~Q

R

Q

D

CLK

## dispositivos de entrada salida.

# UNIDAD III EL NIVEL DE MICROPROGRAMACIÓN.

## III.1. TRAYECTORIA DE LOS DATOS.

## III.2. MICROINSTRUCCIONES.

## III.3. MACROINSTRUCCIONES.

## III.4. CRONOLOGÍA.

## III.5. SECUENCIA.

## III.6. PROCESAMIENTO PARALELO.

# UNIDAD IV MÁQUINA CONVENCIONAL (LENGUAJE DE MÁQUINA).

## IV.1. FORMATO DE INSTRUCCIONES.

## IV.2. DIRECCIONAMIENTO.

## IV.3. TIPOS DE INSTRUCCIONES.

## IV.4. FLUJO DE CONTROL.

# UNIDAD V EL SISTEMA OPERATIVO.

## V.1. MEMORIA VIRTUAL.

### V.1.1 PAGINACIÓN

### V.1.2. SEGMENTACIÓN.

## V.2. INSTRUCCIONES VIRTUALES DE E/S.

## V.3. PROCESAMIENTO PARALELO.

## V.4. CASOS DE ESTUDIO.

### V.4.1. MS-DOS.

### V.4.2. UNÍS.

# UNIDAD VI EL LENGUAJE ENSAMBLADOR.

## VI.1. SEGMENTACIÓN.

## VI.2. DIRECCIONAMIENTO.

## VI.3. TIPOS DE INSTRUCCIONES.

## VI.4. FUNCIONES Y PROCEDIMIENTOS.

## VI.5. MACROINSTRUCCIONES.

## VI.6. LENGUAJES DE ALTO NIVEL.

# UNIDAD VII ARQUITECTURAS AVANZADAS.

## VII.1. PIPELINE.

## VII.2. MÁQUINAS RISC.

## VII.3. ARQUITECTURAS EN PARALELO.

**BIBLIOGRAFÍA.**

* TANENBAUM ANDREW S. “ORGANIZACIÓN DE COMPUTADORAS”, PHH, MÉXICO 1993
* MANO M. MORRIS “ARQUITECTURA DE COMPUTADORAS”, PHH, MÉXICO 1983
* ABEL PETER, “ENSAMBLADOR PARA IBM PC”, PHH, MÉXICO 1996.
* MANO M. MORRIS, “INGENIERIA COMPUTACIONAL”, PHH, MÉXICO 1991
* J. TERRY GODFREY, “LENGUAJE ENSAMBLADOR PARA COMPUTADORAS IBM”, PHH, MÉXICO 1991.