S3 Operaciones Lógicas

Lenin G. Falconí

2024-05-13

Outline

Máquinas de Estados Finitos I

- Modelo computacional que describe un fenómeno
- Es un modelo computacional que describe las transiciones entre diferentes estados posibles.
- Permite inferir el algoritmo o programa a construir para resolver un problema.

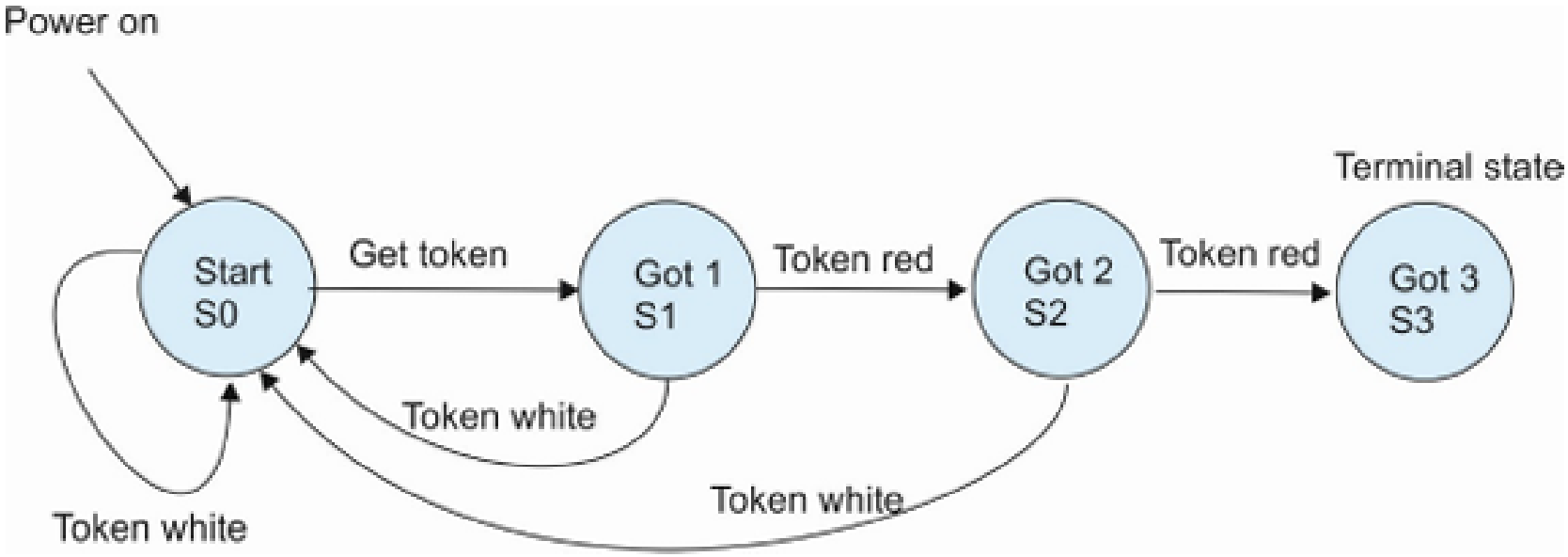


Figure: Máquina de estado Finito para obtener tres tokens Rojos consecutivos

Programa:

- Secuencia de pasos para resolver un problema
- Cada paso ejecuta una operación aritmética o lógica
- Cada operación requiere de un diferente conjunto de señales de control
- Cada operación tiene un código único e.g. ADD, MOVE

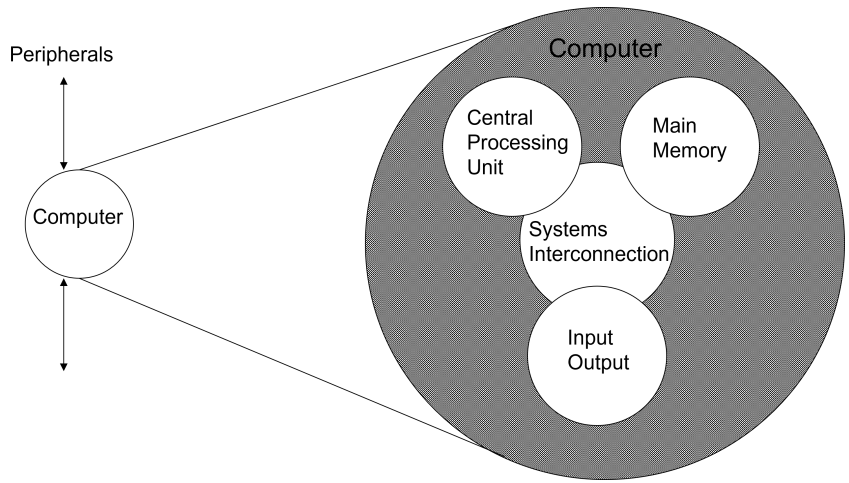
El computador I

- sistema integrado de dispositivos electrónicos que interactúa con el entorno a través de dispositivos periféricos o líneas de comunicación y que procesa información.
- La integración de distintos componentes o subsistemas de manera jerárquica. Este conjunto de sistemas busca realizar las siguientes funciones básicas:
  - Procesamiento de Datos
  - Almacenamiento de Datos
  - Transferencia de Datos
  - Control

La *estructura* estudia cómo están interrelacionados los diferentes componentes del computador. Mientras que el *funcionamiento* estudia la operación de cada componente individual como parte de la estructura.

Estructura Superior del Computador

- El computador interacciona con el medio a través de periféricos o líneas de comunicación.
- CPU: control del funcionamiento del computador y procesamiento.
- Memoria Principal: almacena datos
- I/O: transferencia de datos entre el computador y el entorno externo.
- Sistemas de Interconexión: son los buses de comunicación



CPU

Se encarga del control del funcionamiento del computador y del procesamiento.

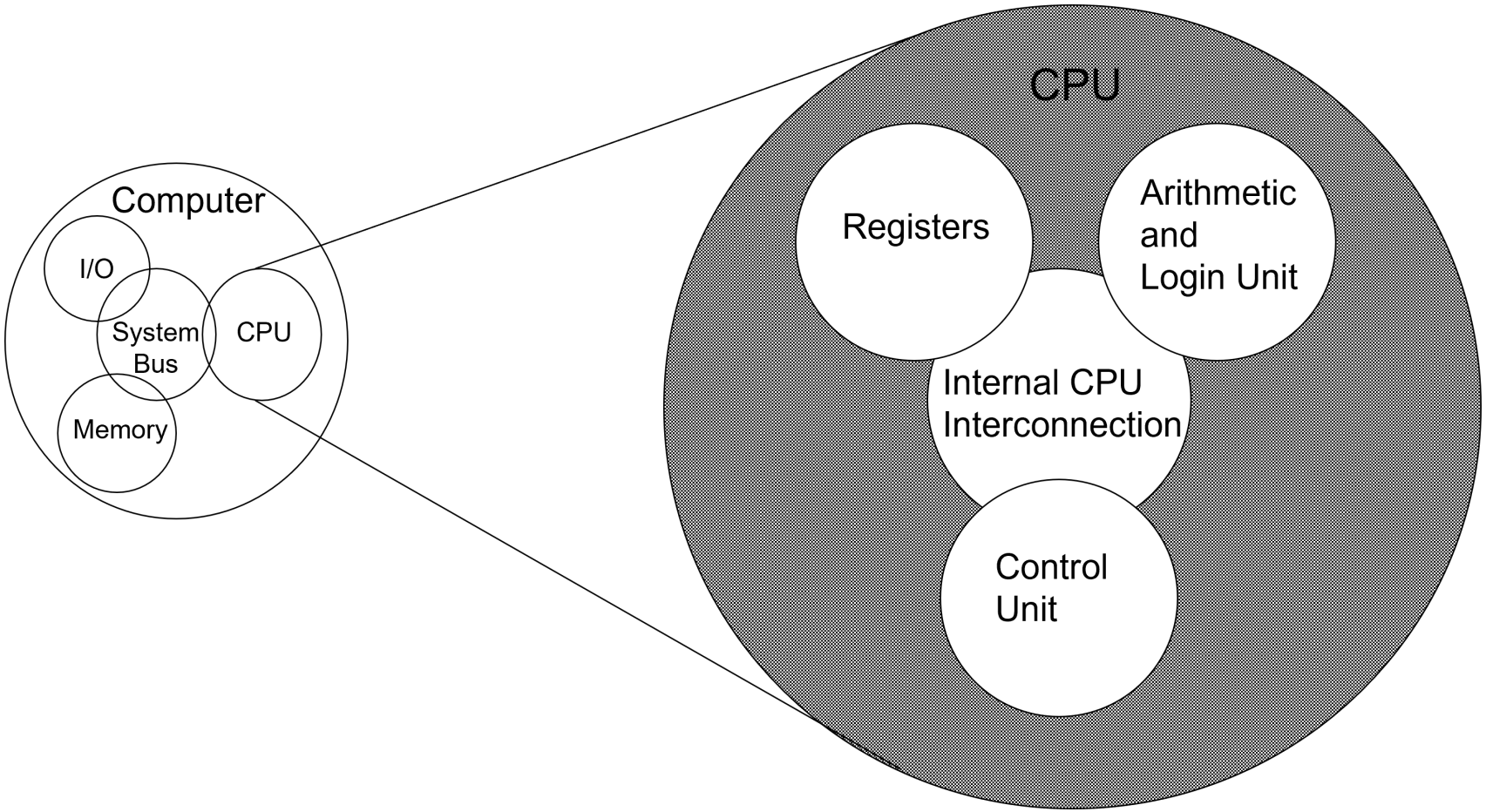


Figure: CPU

Unidad de Control

Conformada por los distintos circuitos digitales, registros, decodificadores y memorias necesarios para el funcionamiento del computador.

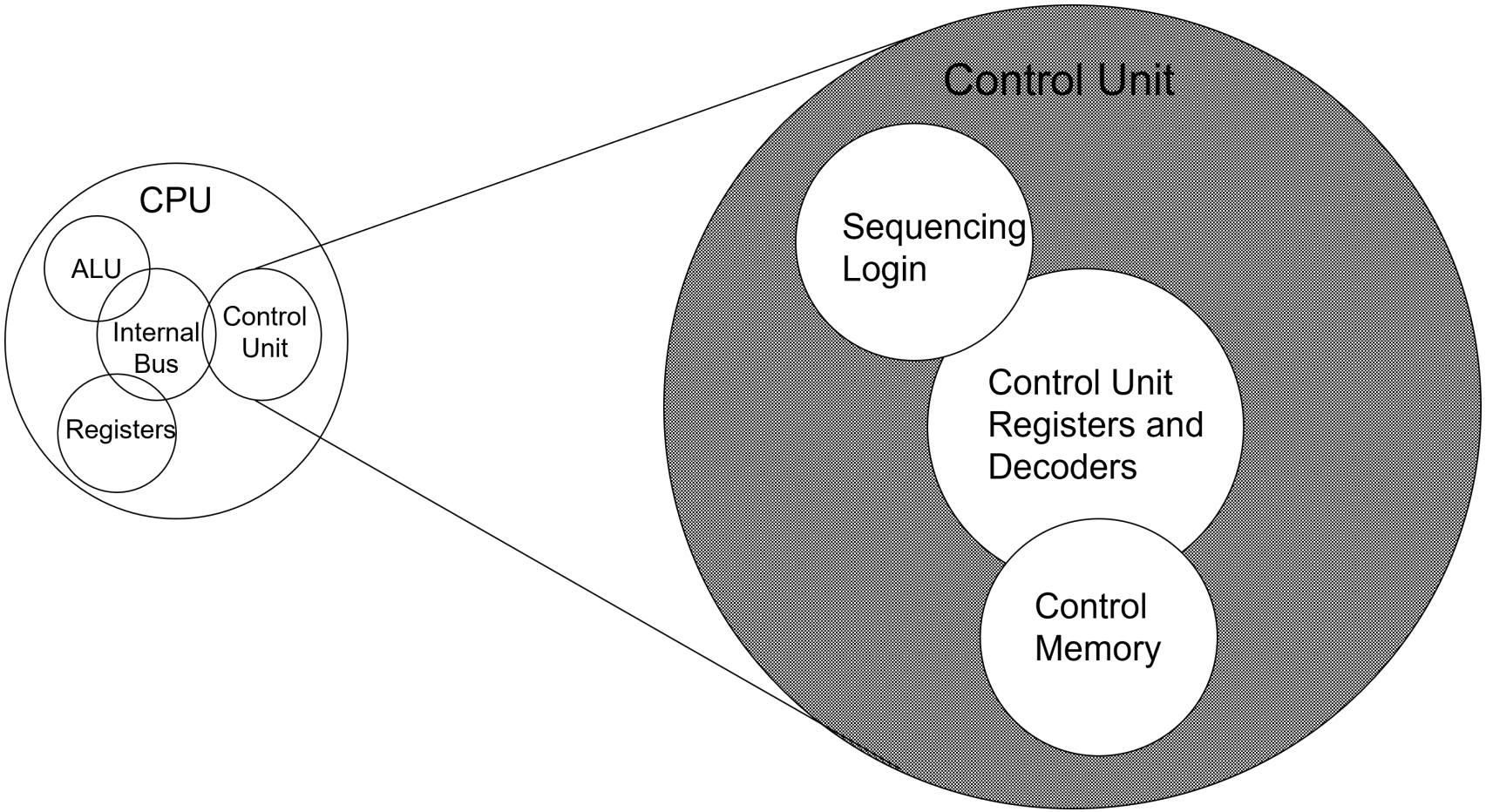
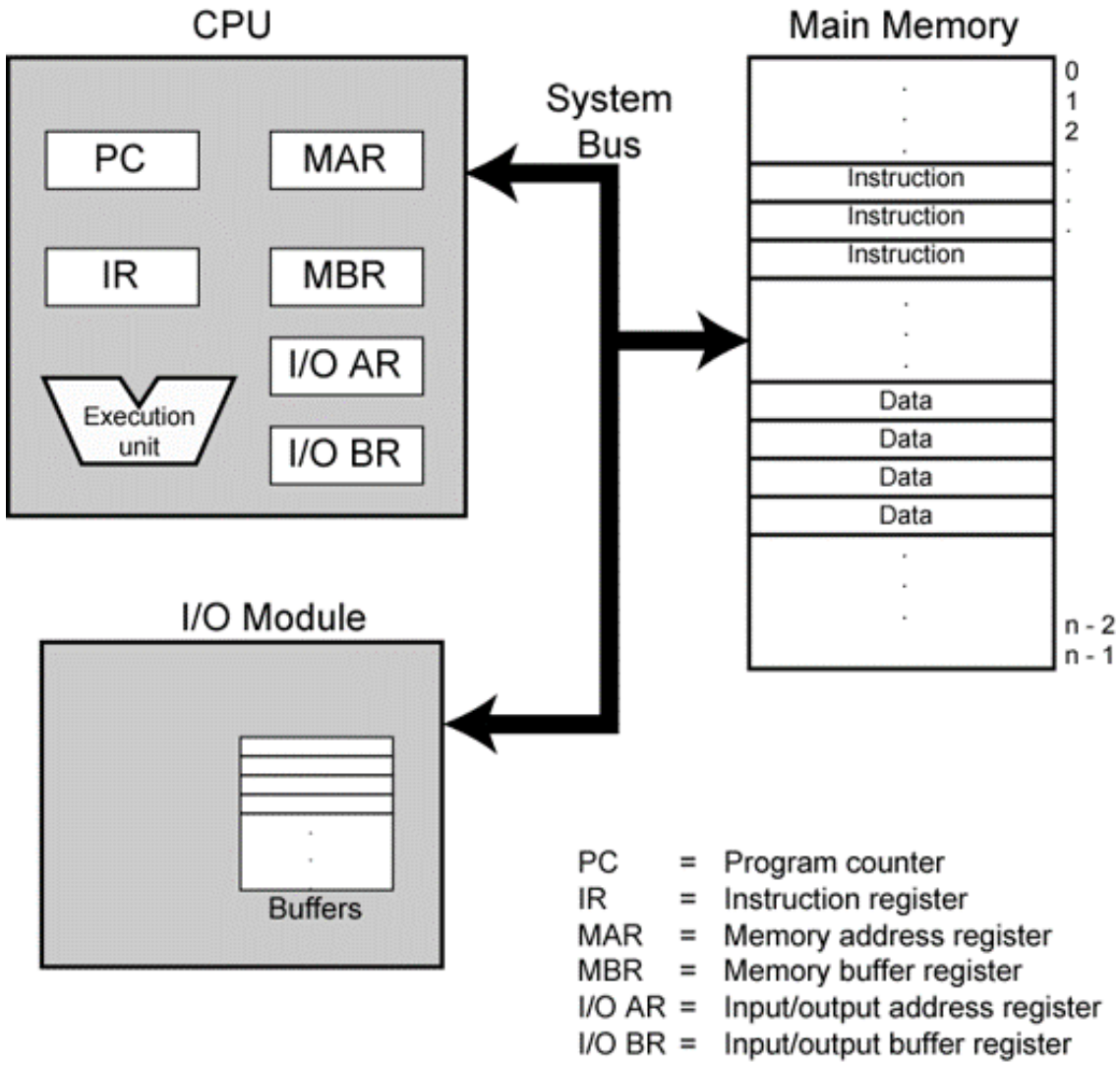


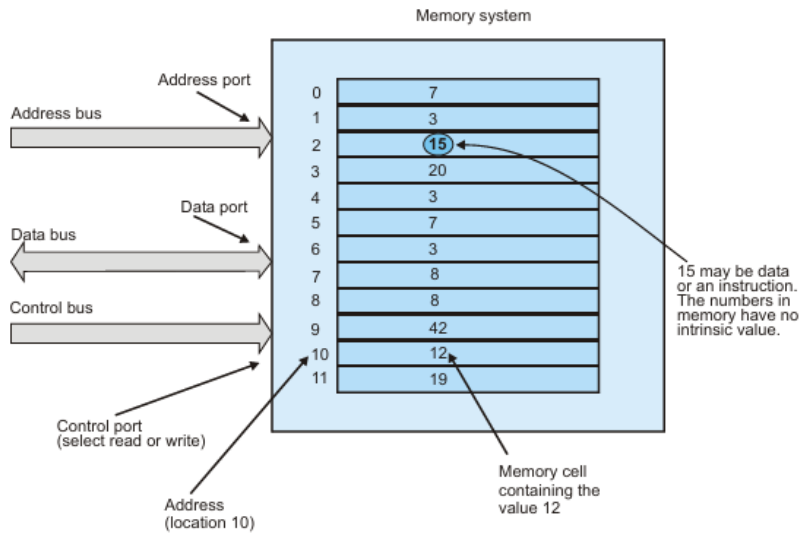
Figure: Unidad de Control

Componentes del Computador



Concepto de Memoria

- Puede ser de escritura o lectura dependiendo de una señal de control
- Las distintas operaciones y datos con los que trabaja el computador son mapeados con direcciones de memoria en donde sus valores se encuentran almacenados.
- Se puede pensar como una lista o tabla de elementos almacenados.



Lenguaje de Transferencia de Registros (RTL) I

- Permite definir de manera sencilla las operaciones en el computador
- No es un lenguaje ensamblador
- No es un lenguaje de Programación
- Es una notación
- Distingue entre las *localidades* de memoria y su *contenido*
- Se usa [ ] para indicar el contenido de una ubicación de memoria
- El símbolo $\leftarrow$ se usa para indicar *transferencia de datos*

1. Suponga una pequeña memoria que tenga 4 bits para el bus de dirección ¿cuántas localidades puede almacenar?
2. Estructure la tabla de memoria suponiendo que el contenido de la memoria será de máximo 8 bits.

Solución

Si las direcciones son de 4 bits, se puede almacenar hasta $2^{n=4} = 16$ localidades.

direcc				dato							
0	0	0	0								
0	0	0	1								
0	0	1	0								
.	.	.	.								
.	.	.	.								
1	1	1	1								

En Hexadecimal tendríamos localidades desde la $0x0$ hasta la $0xF$

Lenguaje de Transferencia de Registros (RTL) I

- $[0x0F] \leftarrow [0x0F] + 1$: el contenido de la localidad de memoria $0x0F$ se incrementa en 1 y se almacena en la misma localidad
- El símbolo $=$ se usa alternativamente para expresar transferencia

Considere las siguientes operaciones:

1. $[0x14] = 5$: el contenido de la dirección de memoria $0x14$ es 5
2. $[0x14] \leftarrow 6$: el valor o literal 6 se carga en $0x14$
3. $[0x14] \leftarrow [6]$: el contenido de la dirección $0x06$ se carga en $0x14$
4. $[0x0C] \leftarrow [0x03] + 3$: el contenido de la dirección $0x03$ se suma con el valor 3 y el resultado se carga en $0x0C$
5. $[0x13] \leftarrow [0x07] + [0x08]$: la suma de los contenidos de las localidades de memoria 7 y 8 se colocan en la dirección 19 (19_{10}=13_{16})
6. $[0x04] \leftarrow [[0x02]]$: puntero o direccionamiento indirecto. El valor a copiar en la localidad 4 es el contenido en la dirección definida por el contenido de la localidad 2.

Ejercicio I

Considere la siguiente memoria abstracta.
Obtenga: $X =$
$3+[0x04]+[1+[0x03]]+[[0x0A]]+[[0x09]*3]$

Dirección	Dato
0x00	6
0x01	2
0x02	3
0x03	4
0x04	5
0x05	2
0x06	8
0x07	1
0x08	5
0x09	2
0x0A	1
0x0B	5

Ejercicio - Solución I

Considere la siguiente memoria abstracta.
Obtenga: $X =$
$3+[0x04]+[1+[0x03]]+[[0x0A]]+[[0x09]*3]$
$X = 3 + 5 + 2 + 2 + 8$

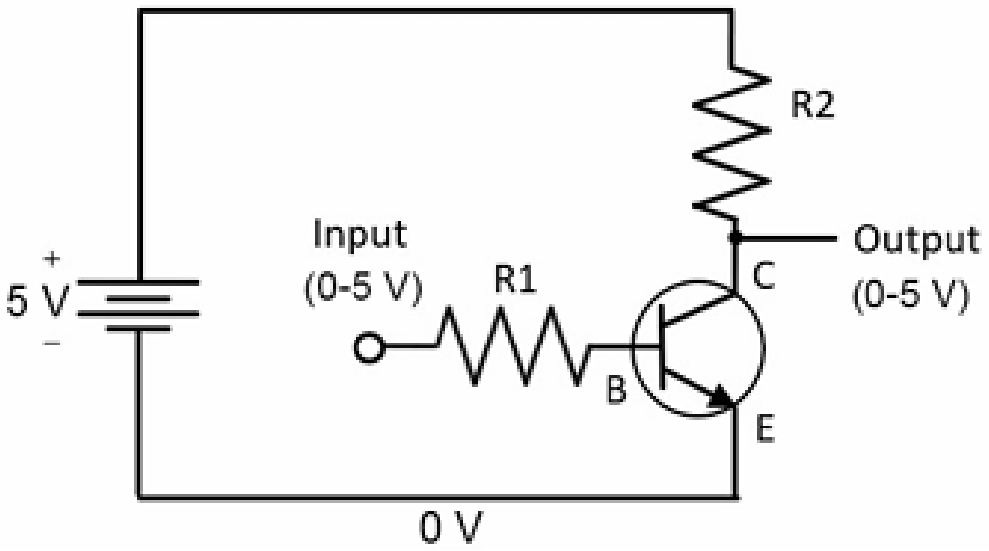
Dirección	Dato
0x00	6
0x01	2
0x02	3
0x03	4
0x04	5
0x05	2
0x06	8
0x07	1
0x08	5
0x09	2
0x0A	1
0x0B	5

Lógica Digital - Circuito Eléctrico I

- Los materiales conductores tienen la característica de producir una corriente eléctrica en presencia de un campo eléctrico.
- El voltaje V, la corriente I y la resistencia R se relacionan con la Ley de Ohm $V = IR$
- Un semiconductor es un material que exhibe las características tanto de un buen conductor como de un buen aislante. Esta característica se controla por una entrada de control.
- Un transistor es un semiconductor que opera como un switch digital. Cambia de alta a baja resistencia dependiendo del estado de una señal de entrada.

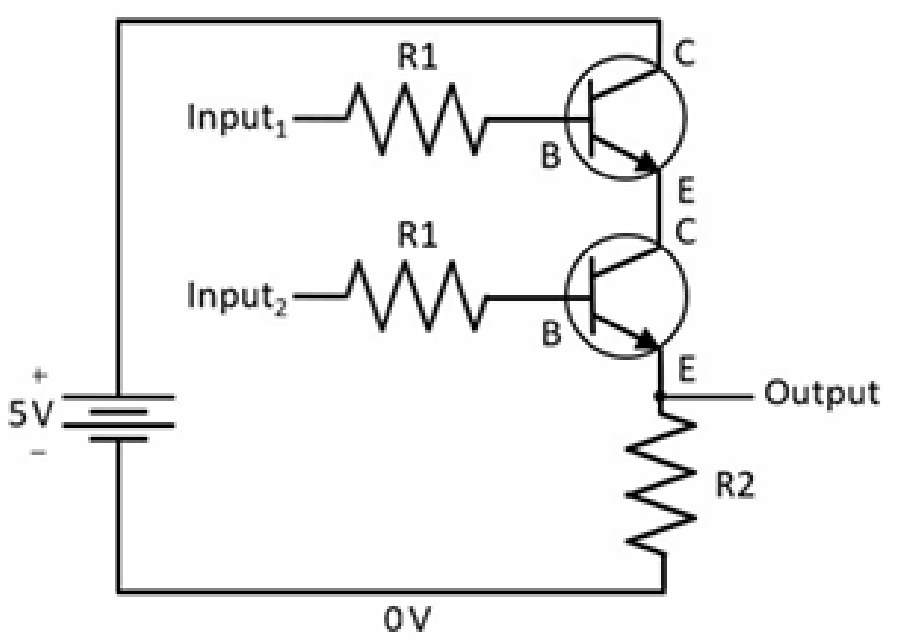
Compuertas Lógicas

- Son arreglos de circuitos con transistores que permiten realizar operaciones lógicas
- Un transistor tiene un voltaje de switching de 0.7V.
- Con un $V \geq 0.7$, el transistor se activa y la resistencia entre colector y emisor se reduce, colocando la salida a un bajo voltaje.
- El comportamiento del circuito se puede expresar en una tabla de verdad



Compuertas Lógicas

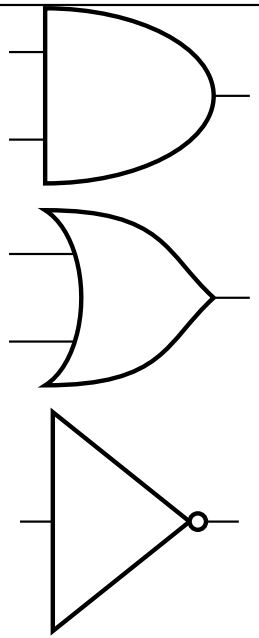
$input_1$	$input_2$	salida
0	0	0
0	1	0
1	0	0
1	1	1



Álgebra de Boole y Compuertas Lógicas I

- Utilizada para resolver problemas de diseño de circuitos de conmutación
- Las variables y las operaciones son lógicas
- 1 equivale a Verdadero
- 0 equivale a Falso
- Las operaciones lógicas AND, OR y NOT se denotan como:

AND	$A \wedge B$	$A \cap B$	$A \cdot B$
OR	$A \vee B$	$A \cup B$	$A + B$
NOT	$\neg A$	$A^{\complement}$	$\bar{A}$



Álgebra de Boole y Compuertas Lógicas II

- Es importante notar que las compuertas NAND y NOR son las respectivas negaciones de las compuertas AND y OR i.e.

$$A\,NAND\,B = \neg(A \wedge B) = \overline{A \wedge B}$$

$$A\,NOR\,B = \neg(A \vee B) = \overline{A \vee B}$$

- AND, OR y NOT son un conjunto funcionalmente completo.
- NAND y NOR pueden implementar cualquier circuito digital ya que las AND, OR y NOT se pueden implementar directamente sólo con compuertas NAND o NOR. Condición favorable para procesos de fabricación.

Circuitos Combinacionales

- Conjunto de compuertas lógicas interconectadas cuya salida, en un momento dado, es función únicamente de las entradas en ese instante.
- La relación puede ser expresada por *funciones booleanas* o por *tablas de verdad*.
- La ecuación booleana se puede simplificar con aplicación de las identidades o postulados básicos del álgebra booleana o por Mapas de Karnaugh
- Se pueden expresar como Suma de Productos (SOP) o productos de sumas (POS)
- El Teorema de Morgan permite hacer la conmutación de las dos representaciones.

Representación de Min-Terms o Sumas de Productos (SOP)

Sea $F(X_1, X_2, \cdot, X_n)$ la salida de un circuito lógico combinacional booleano que recibe como entradas $X_1, X_2, \cdot, X_n$, encontes:

1. Localizar los casos de la *Tabla de Verdad* donde la Función $F = 1$
2. Para cada uno de los casos identificados escribir el producto de las entradas considerando que si la entrada en la tabla vale 1, se mantiene el símbolo. Si vale 0, se escribe el complemento.
3. Sume los productos obtenidos

Representación de Max-Terms o Productos de Sumas (POS)

Sea $F(X_1, X_2, \cdot, X_n)$ la salida de un circuito lógico combinacional booleano que recibe como entradas $X_1, X_2, \cdot, X_n$, encontes:

1. Localizar los casos de la *Tabla de Verdad* donde la Función $F = 0$
2. Para cada uno de los casos identificados escribir la suma de las entradas considerando que si la entrada en la tabla vale 0, se mantiene el símbolo. Si vale 1, se escribe el complemento.
3. Sume los productos obtenidos

Ejemplo de Representación como SOP I

Considera la Siguiente Tabla de Verdad:

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

- Los casos que interesan son: 000, 010, 011, y 110, porque $F = 1$.
- En consecuencia, existen 4 Sumas de Productos. En cada producto, si la variable está con 0 se complementa. Si está con 1 se deja:

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

- Una vez obtenida se debe reducir por medio de Mapa K. o postulados del álgebra booleana.

Ejemplo de Representación como POS I

Considera la Siguiente Tabla de Verdad:

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

- Los casos que interesan son: 001, 100, 101, y 111, porque $F = 0$
- En consecuencia, existen 4 Productos de Sumas. En cada producto, si la variable está con 1 se complementa. Si está con 0 se deja:

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

- Una vez obtenida se debe reducir por medio de Mapa K. o postulados del álgebra booleana.

Circuitos Secuenciales

La salida actual de estos circuitos depende de la entrada actual y de la historia pasada de las entradas. Estos circuitos usan una señal de reloj, generalmente. Ejemplos son:

- Biestables o latch SR
- Biestable D
- Registros
- Contadores

Ejercicios I

1. A partir de la tabla de verdad de la compuerta OR exclusiva de dos entradas obtenga la función booleana como SOP (min-términos).
2. Para el ejercicio anterior obtenga la representación en POS (max-términos).
3. ¿Puede representar el circuito sólo con compuertas NAND?
4. Simplificar $F = ACD + \bar{A}BCD$. Resp:$CD(A + B)$
5. Simplificar $F = ABC + A\bar{B}\overline{\bar{A}\bar{C}}$ R: $A(\bar{B} + C)$
6. A partir de la Tabla 1 de verdad obtener la representación en SOP.
7. Usando Mapas de Karnaugh obtenga la simplificación del circuito de la Tabla 1

Ejercicios II

Table: Ejercicio de tres variables

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