

Unidad 1:

Introducción a las máquinas de estado finito

Contenidos:

- Introducción
- Máquinas de Estados Finitos (FSM)
- Tipos de FSM
- Implementación de FSM
 - Lógica discreta
 - En Registro - Memoria

1. Introducción

- **Conjunto:** Colección de objetos relacionados entre sí. Un conjunto sólo indica los elementos que lo componen, por tanto no es necesario tener un orden, así como tampoco se repiten los elementos.

Un conjunto finito es aquel que podemos contar sus elementos como:

$$C = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$$

Claramente vemos que son 4 elementos.

En un conjunto infinito encontramos:

- Los conjuntos infinitos no numerables, que son aquellos que no podemos contar sus elementos
 - Los conjuntos infinitos numerables, que son aquellos que podemos numerar sus elementos.
- **Subconjunto:** Colección de elementos que a su vez pertenece a otro conjunto de igual o mayor

número de miembros, por ejemplo:

Considere:

$$A = \{a, b\}, \quad B = \{z, a, c, b, d\}$$

A es subconjunto de B: $A \subset B$

Si A tiene los mismos elementos y es subconjunto de B, se escribe: $A \subseteq B$.

- **Símbolo:** Representación abstracta de un objeto, ya sea un dígito o letra. En general representa un elemento, como $a, 1, \pi$, etc.
- **Alfabeto:** También llamado vocabulario, es un conjunto finito de símbolos. Debe existir al menos un símbolo en el alfabeto. Se le llama comúnmente con el símbolo Σ a este conjunto.
- **Cadena:** Secuencia de símbolos hecha con los elementos de un alfabeto. Por eso se dice que una cadena omega (ω) sobre un alfabeto sigma universal (Σ^*), es decir, $\omega \in \Sigma^*$ donde Σ^* es un conjunto formado de todas las cadenas posibles que se puedan hacer con los elementos de Σ .
Una cadena no es un subconjunto del alfabeto universal, es alguno de sus elementos. Una cadena no puede ser infinita. También se conoce como frase o palabra. Por ejemplo, consideremos:

$$\Sigma = \{1, 0\} \text{ entonces}$$

$$\Sigma^* = \{1, 0, 00, 01, 10, 11, 000, \dots\}$$

Una cadena sobre sigma ω podría ser:

101011010

Esta cadena es un elemento de Σ^* .

- **Lenguaje:** Conjunto de cadenas, las cuales deben estar formadas con los símbolos de un alfabeto Σ , entonces decimos que el lenguaje L está sobre el alfabeto Σ .

Por ejemplo:

El lenguaje $L = \{100, 001, 001, 111\}$ se forma con los elementos de $\Sigma = \{0, 1\}$.

La cadena $\sigma = 1010$ se forma con los elementos de Σ , pero no pertenece a L y se denota como $\sigma \notin L$.

- **Estado:** Es la situación o las condiciones en que se halla un objeto en algún momento. Dicho objeto no puede estar en más de un estado al mismo tiempo.

En el caso de una máquina, son las características que posee en un momento dado.

- **Algoritmo:** Secuencia finita de instrucciones bien definidas. Se compone de una sucesión de pasos que llevan siempre a un mismo resultado.

- **Computación:** La aplicación de un algoritmo sobre un conjunto de datos de entrada y obteniendo otro conjunto de datos de tal proceso.

2. Máquinas de Estados Finitos (FSM)

Las máquinas de estados finitos conocidas como FSM nos sirven para realizar procesos bien definidos en un tiempo discreto. Reciben una entrada, hacen un proceso y nos entregan una salida. Notemos que estas máquinas realizan una computación.

Sólo se puede leer un dato en cada paso que se realice. Cada entrada diferente genera una salida diferente, pero el mismo resultado con los mismos datos de entrada. Una computación es capaz de resolver un problema si y sólo si tiene una solución algorítmica.

Una FSM es un modelo abstracto para la manipulación de símbolos que nos permiten saber si una cadena o lenguaje nos puede generar otro conjunto de símbolos como resultado. Llamaremos una FSM como un autómata finito.

Los autómatas se caracterizan por tener un estado inicial, reciben una cadena de símbolos, cambian de estado por cada elemento leído o pueden permanecer en el mismo estado. También pueden contener un conjunto de estados finales o aceptables que indican si una cadena pertenece al lenguaje al final de una lectura.

3. Tipos de FSM

Los autómatas se clasifican en:

- Autómatas finitos deterministas
- Autómatas finitos no deterministas

Autómatas finitos deterministas

Uno de estos recibe en secuencia una cadena de símbolos y cambia de estado por cada símbolo leído o también puede permanecer en el mismo estado. Al final de la lectura, el estado del autómata nos indica si la cadena pertenece al lenguaje que describe la máquina.

Las partes de un autómata son 5:

$$A = \{Q, q_0, F, \Sigma, \delta\}$$

donde:

Q es un conjunto finito de estados

q_0 es el estado inicial donde $q_0 \in Q$ y debe haber uno y sólo un estado inicial

F es un conjunto de estados finales donde $F \subseteq Q$ y q_0 también puede ser estado final

Σ es un alfabeto finito de entrada

δ es la función de transición que $Q \times \Sigma \rightarrow Q$

Supongamos que el autómata está en el estado q_i donde $q_i \in Q$, tenemos a "A" donde "A" $\in \Sigma$. Una entrada "A" causa que el autómata

cambie de q_i a q_k .

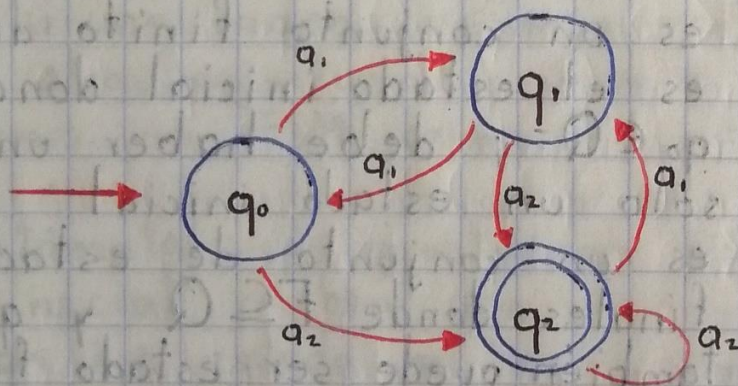
La función Δ llamada función de transición describe este cambio de la forma:

$$\delta(q_i, a) \rightarrow q_k$$

De esta forma obtenemos un nuevo estado.

Se entiende por transición como el proceso que hace.

La forma más fácil de imaginarnos un autómata es mediante un diagrama de transición. Un diagrama de transición es un digrafo etiquetado por los elementos de un autómata para este caso, pero se puede representar cualquier máquina de estados.



En un diagrama de transición existe un nodo por cada estado q_i de Q . Los estados finales están representados con un doble círculo. El estado inicial q_0 es apuntado por una flecha que no proviene de ninguno de los estados.

Para cada estado q_i y un símbolo a hay exactamente una y sólo una flecha que inicia en q_i y termina en $\delta(q_i, a)$, es decir q_k . La flecha es etiquetada como a . Si q pertenece a F decimos que la entrada es aceptada.

Debe haber una flecha saliendo de cada estado por cada símbolo $a_0, a_1, a_2, \dots, a_n$; así, todos los estados tienen el mismo número de flechas saliendo de cada uno de ellos. De este modo garantizamos que el autómata puede ser llamado determinista, pues siempre hay una transición definida.

Para descubrir por completo una función Δ ocupamos una tabla de transición. Las columnas se etiquetan con los símbolos de entrada. Las filas son etiquetadas con los estados y en las intersecciones se colocan los nuevos estados $\delta(q_i, a)$, suponiendo que $q_i \in Q$ y $a \in \Sigma$.

Entonces, la tabla de transición para la figura anterior es la siguiente:

	a_1	a_2
$\rightarrow q_0$	q_1	q_2
q_1	q_0	q_2
$\leftarrow q_2$	q_1	q_2

El estado inicial tiene una flecha que apunta a él. Los estados finales tienen una flecha que salen de ellos. Los demás no tienen una flecha.

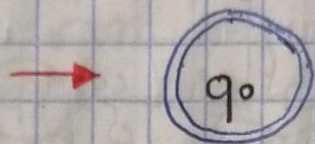
Una tabla de transición representa una función δ que recibe un símbolo y un estado. Si queremos introducir una cadena omega que pertenece al alfabeto universal Σ^* , donde el alfabeto universal es la cerradura del alfabeto (es decir, sólo Σ), en lugar de un solo símbolo, usaremos δ^* conocida como función de transición extendida que nos permite manejar una cadena dado que es una función $Q \times \Sigma$ y cumple con:

- $\delta^*(q_i, a) \rightarrow \delta(q_i, a)$ donde $q_i \in Q$ y $a \in \Sigma$
- $\delta^*(q_i, \epsilon) \rightarrow q_i$ donde $\epsilon \in \Sigma$ es el elemento vacío
- $\delta^*(q_i, w) \rightarrow \delta^*(\delta(q_i, a), w)$ donde $a \in \Sigma$ y $w \in \Sigma^*$

El autómata permanece en el mismo estado al introducir ϵ , es decir, no cambia de estado. Se comporta como un símbolo neutro para δ^* .

Épsilon puede ser aceptado si y sólo si q_f también pertenece al conjunto de estados finales.

El autómata que sólo acepta el elemento vacío es:



Otra propiedad de δ^* nos define cómo evaluar cadenas de forma recursiva. Se toma el primer símbolo de la cadena y se evalúa con δ . El estado resultante es evaluado con el símbolo para obtener un nuevo estado y así sucesivamente teniendo:

$$\delta(q_0, a_0), \delta(\delta(q_0, a_0), a_1), \dots$$

Hasta terminar de evaluar la cadena. Si el autómata se encuentra en un estado final f se entiende que:

$$\delta^*(q_0, w) = f$$

Y la cadena es aceptada.

Formalmente decimos que un lenguaje L es aceptado por un autómata A si y sólo si $\exists w \in \Sigma^*$ y cumple con la función de transición extendida.

Concluimos que:

$$L(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$$

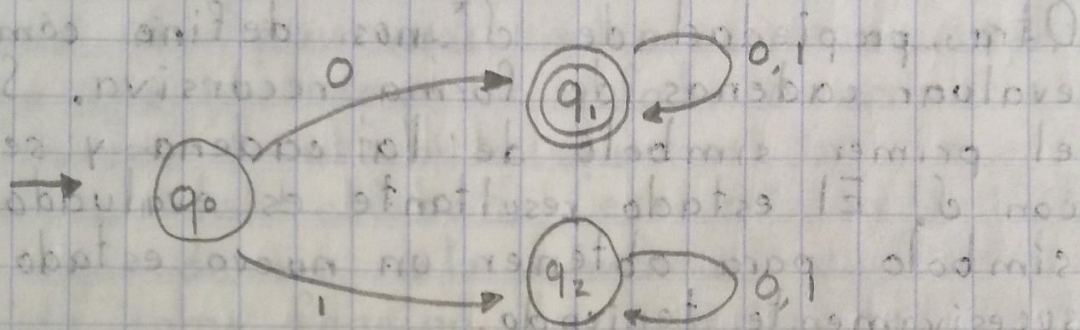
Ejercicio: Obtenga un AFD que den el conjunto de cadenas que inician en 0. Utilice el alfabeto siguiente:

$$\Sigma = \{0, 1\}$$

Nuestro autómata se define como:

$$A = (Q, q_0, F, \Sigma, \delta)$$

q_0 = estado inicial



$$Q = \{q_0, q_1, q_2\} \quad \delta(q_0, 0) = q_1$$

$$\delta(q_0, 1) = q_2$$

$$F = \{q_1\}$$

$$\delta(q_1, 0) = q_1$$

$$\delta(q_1, 1) = q_1$$

$$\delta(q_2, 0) = q_2$$

$$\delta(q_2, 1) = q_2$$

$$\hat{\Gamma} = (w, q_0)$$

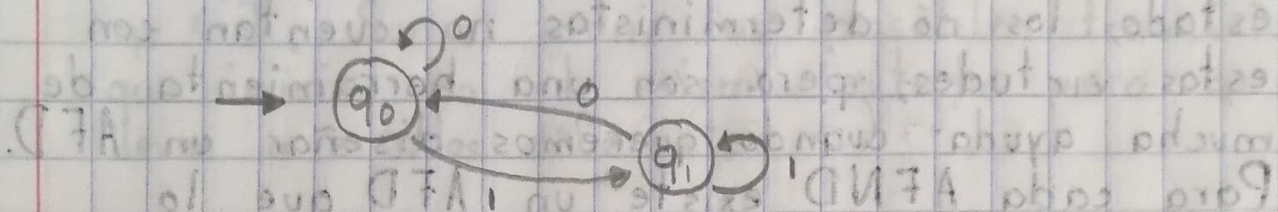
Determine si alguna de las siguientes cadenas son aceptadas por el autómata diseñado.

a) $w_1 = 1100101$ No, porque $\delta^*(q_0, w_1) = q_2 \neq q_1$ X

b) $w_2 = 001001$ Sí, porque $\delta^*(q_0, w_2) = q_1$ ✓

$$\hat{\Gamma} = (w, q_0) \text{ o } \hat{\Gamma} = (w, q_1)$$

Ejercicio: Obtenga un AFD que dé el conjunto de cadenas que terminan en 1 con $\Sigma = \{0, 1\}$



$Q = \{q_0, q_1\}$

$F = \{q_1\}$

$\delta(q_0, 0) = q_0$

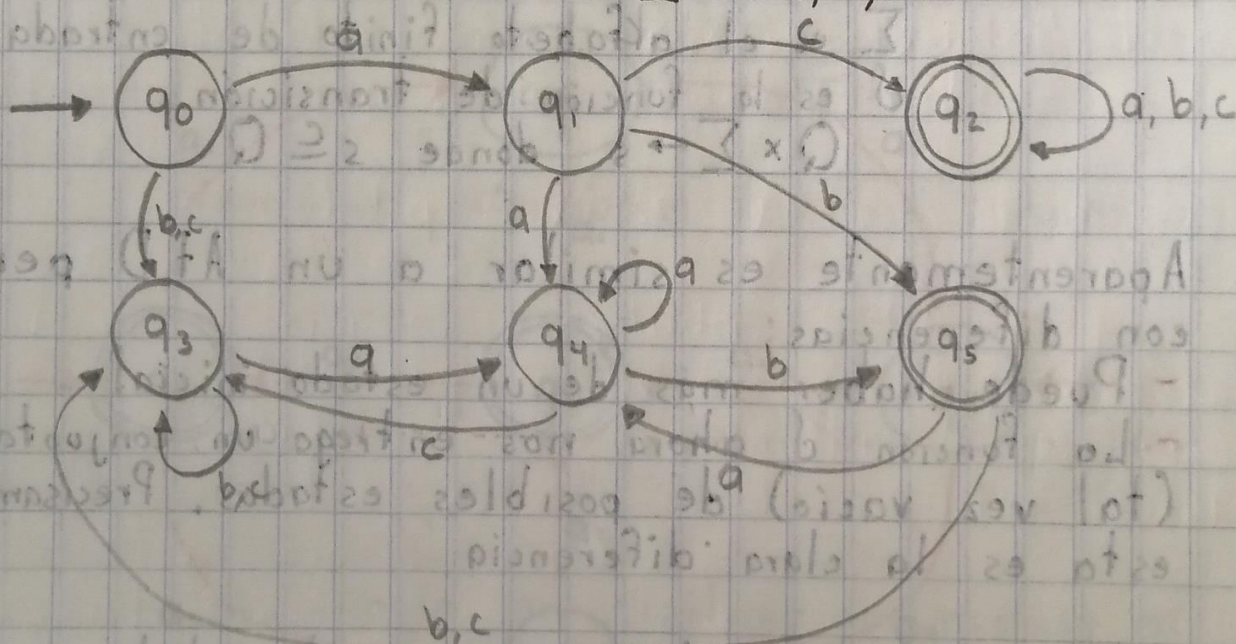
$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_0$

$\delta(q_1, 1) = q_1$

Ejercicio: Realice un AFD que obtenga el conjunto de cadenas que inician en la subcadena "ac" o que terminen en la subcadena "ab".

Utilice el alfabeto $\Sigma = \{a, b, c\}$



Autómatas finitos no deterministas

Si con los AFD sólo se puede llegar de una forma de un estado a otro y con un estado, los no deterministas no cuentan con estas virtudes, pero son una herramienta de mucha ayuda cuando queremos diseñar un AFD. Para cada AFND existe un AFD que lo representa y que acepta el mismo lenguaje.

Definimos un AFND como:

$$A = \{Q, I, F, \Sigma, \delta\}$$

donde:

Q es un conjunto finito de estados

I es un conjunto de estados iniciales

donde $I \subseteq Q$

F es un conjunto de estados finales

donde $F \subseteq Q$

Σ es el alfabeto finito de entrada

δ es la función de transición

$$Q \times \Sigma \rightarrow S \text{ donde } S \subseteq Q$$

Aparentemente es similar a un AFD pero con diferencias:

- Puede haber más de un estado inicial
- La función δ ahora nos entrega un conjunto (tal vez vacío) de posibles estados. Precisamente esta es la clara diferencia

Cuando todas las transiciones están determinadas en un autómata, se tiene un AFD. Si se tiene al menos una transición no definida, éste es un AFND.

En la función de transición extendida δ^* también hay cambios. Se puede comenzar en cualquiera de los estados iniciales q_i donde $q_i \in I$. Suma todas las posibles transiciones que inicien en el mismo estado pero que lleven a estados diferentes.

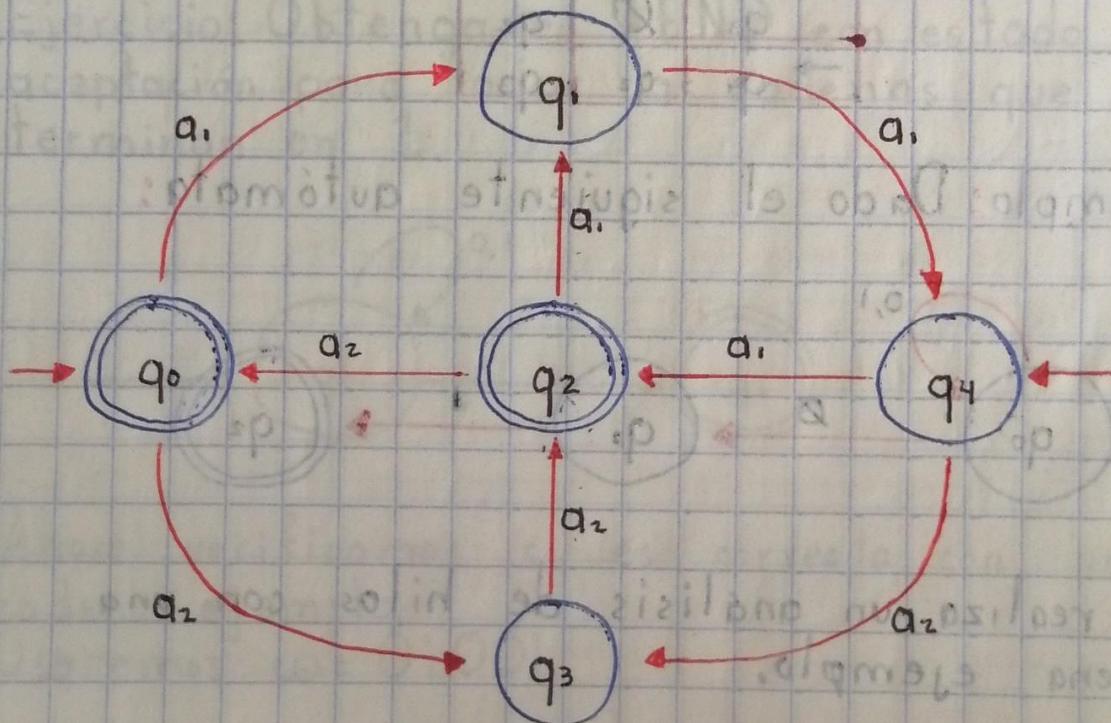
Entonces podemos definir lo siguiente:

$$\delta^*(q_i, a\omega) = \sum_{i=0}^n \delta^*(t_i, \omega)$$

Se dice que una cadena es aceptada si:

$$\delta^*(q_i, a\omega) = q_f \text{ donde } q_f \in F$$

Observemos:



Vemos que existe más de un estado inicial y las transiciones siguientes no están determinadas:

$$\delta(q_1, a_2)$$

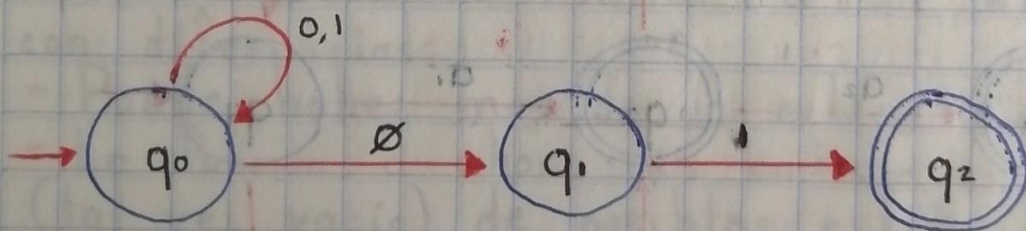
$$\delta(q_3, a_1)$$

Basta una de las condiciones anteriores para considerar al autómata como no determinista.

Su tabla de transición es la siguiente:

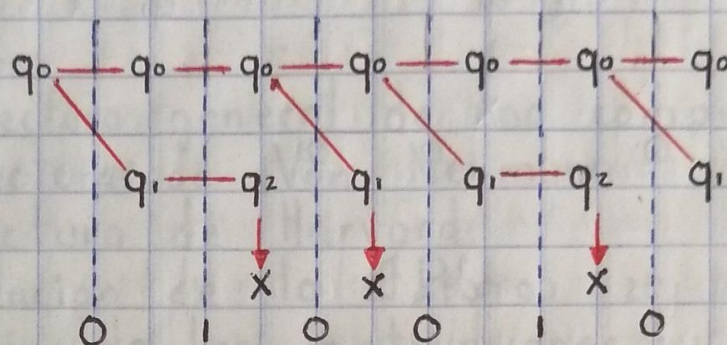
	a_1	a_2
$\leftrightarrow$ q_0	q_1	q_3
q_1	q_4	$\emptyset$
$\leftarrow$ q_2	q_1	q_4
q_3	$\emptyset$	q_2
$\rightarrow$ q_4	q_2	q_3

Ejemplo: Dado el siguiente autómata:



Se realiza un análisis de hilos con una cadena ejemplo.

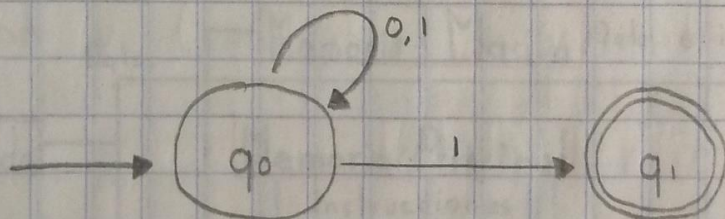
Con $w = 010010$ el análisis de hilos es el siguiente:



Esta cadena no pertenece al AFND porque ninguno de los estados después del último hilo es q_2 , es decir, el estado final. Recordemos que:

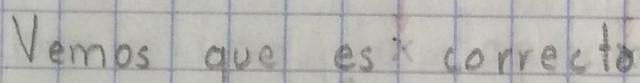
$$L(A) = \{w \mid \delta^*(q_0, w) \cap F = \emptyset\}$$

Ejercicio: Obtenga el AFND con estado de aceptación para todas las cadenas que terminan en 1.



Ahora verificamos si es correcto con una cadena ejemplo.

Usaremos $w = 01001$


$$B = \{ \cap (0, 5) \} \text{ ist } = (A)$$

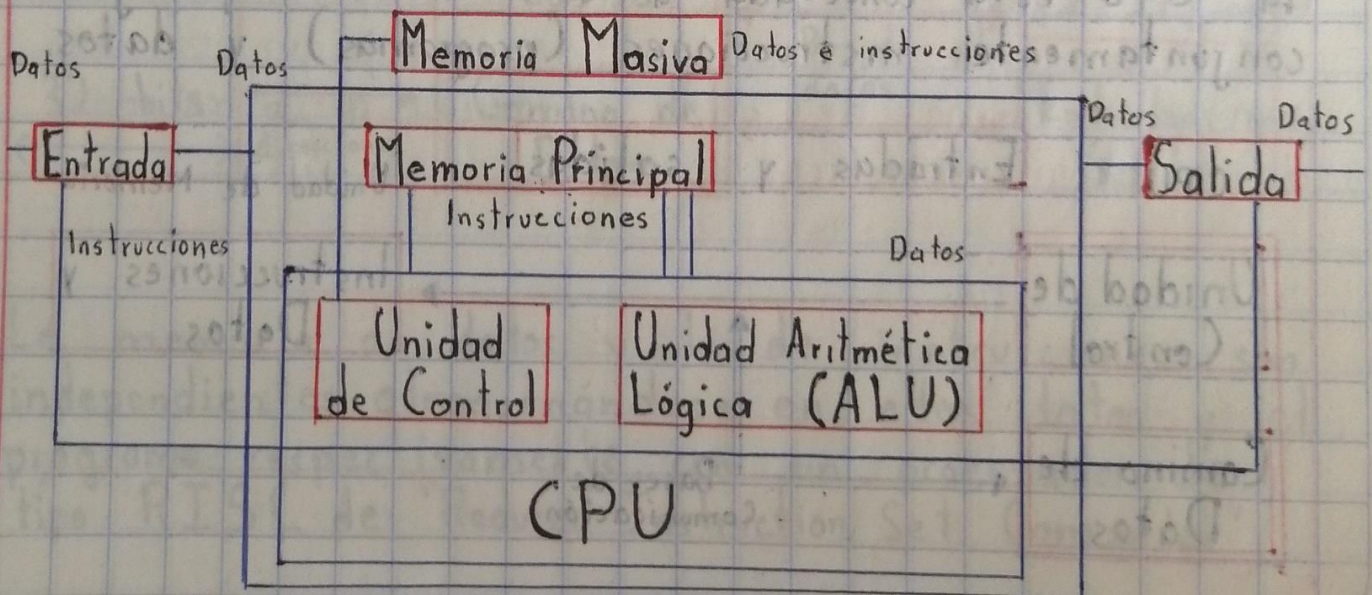
30. Observe que a função $f(x) = x^2 + 2x + 1$ pode ser escrita como $f(x) = (x+1)^2$. Assim, a função $f(x)$ é sempre não negativa, ou seja, $f(x) \geq 0$ para todo x real.

Unidad 2: Organización de las computadoras

Contenidos:

- Arquitectura general de una computadora
- Arquitectura de Von Neumann
- Arquitectura de Harvard
- Organización de la CPU
- Ejecución de las instrucciones
- La CPU como una FSM
 - Ciclo de fetch
- Pipeline
- Predicción de saltos
- Predictores de N bits con historia
- Concepto de CISC, RISC
- Organización de datos en memoria
 - Big endian
 - Little endian

1. Arquitectura General de una Computadora



El desarrollo de los computadores y de la informática se liga al de la electrónica. El avance tecnológico y el uso de computadores la ha consolidado como disciplina de ingeniería.

Las dos principales causas de la problemática en la arquitectura de computadores fueron:

- La independencia entre el hardware y el software
- El seguimiento a ultranza de la arquitectura Von Neumann

Pero hoy en día los inconvenientes que enfrentamos son:

- El factor tiempo
- El alcance de límites difícilmente superables de la tecnología hardware
- Límite de coste y ventas

2. Arquitectura de Von Neumann (Clásica)

John Von Neumann propuso una arquitectura donde la CPU está conectada a una memoria que guarda conjuntamente instrucciones (programas) y datos.



Esta se denomina tipo CISC de "Complex Instruction Set Computer". Las instrucciones complejas exigen mucho tiempo de CPU.

Por tanto tiene dos ventajas:

- El procesador se ve obligado a hacer varios accesos a memoria a instrucciones complejas
- La operación y su velocidad están limitadas por el efecto cuello de botella que es que un bus único para datos e instrucciones impide superponer ambos accesos.

3. Arquitectura de Harvard (Moderna)

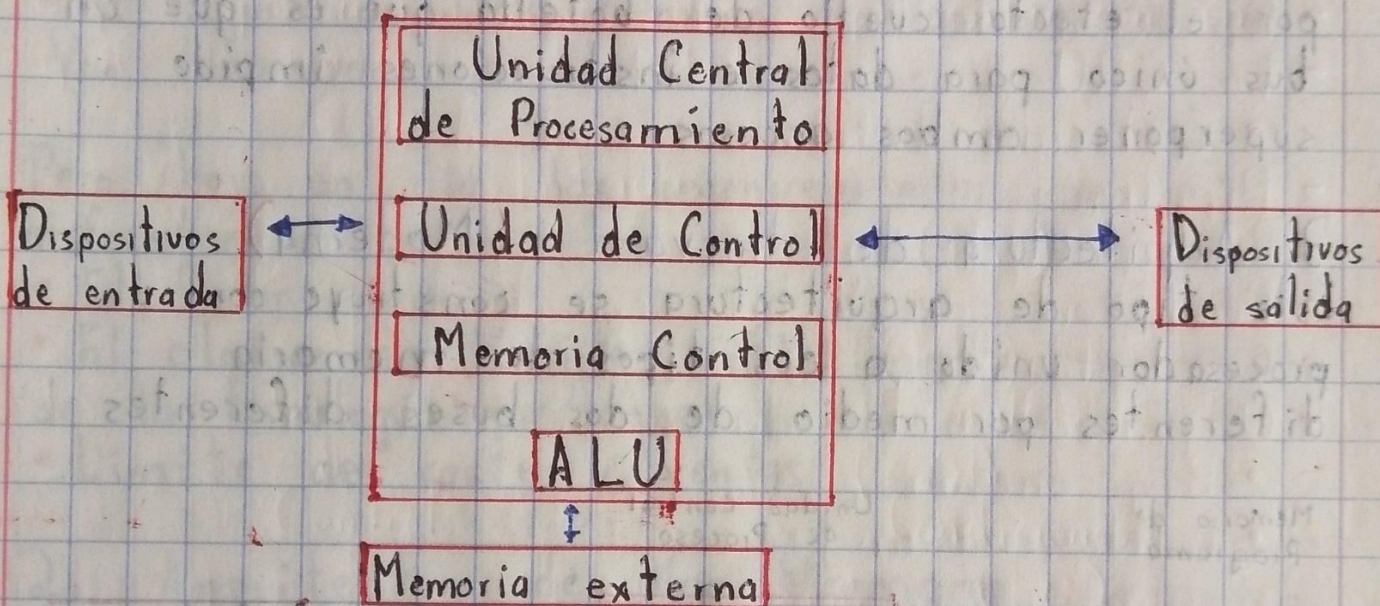
Este tipo de arquitectura se construye con un procesador unido a dos tipos de memoria diferentes por medio de dos buses diferentes



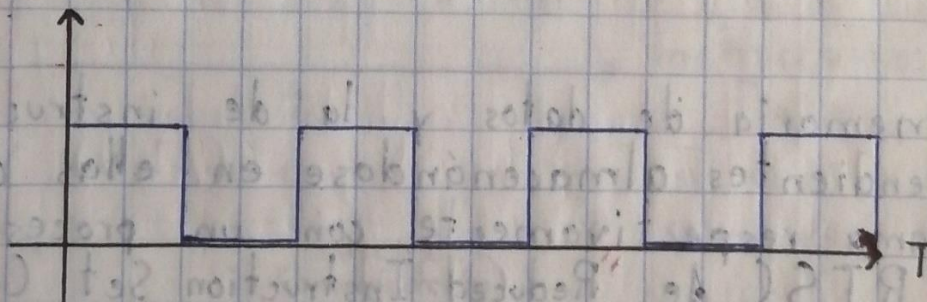
La memoria de datos y la de instrucciones son independientes almacenándose en ellas datos y el programa respectivamente con un procesador de tipo RISC de "Reduced Instruction Set Computer".

4. Organización de la CPU

La organización básica de una computadora consiste en la unidad de entrada, por la cual se introducen datos e instrucciones; la CPU donde se procesan los datos de acuerdo a las instrucciones dadas; y la unidad de salida por la que se presenta la información resultante al usuario.



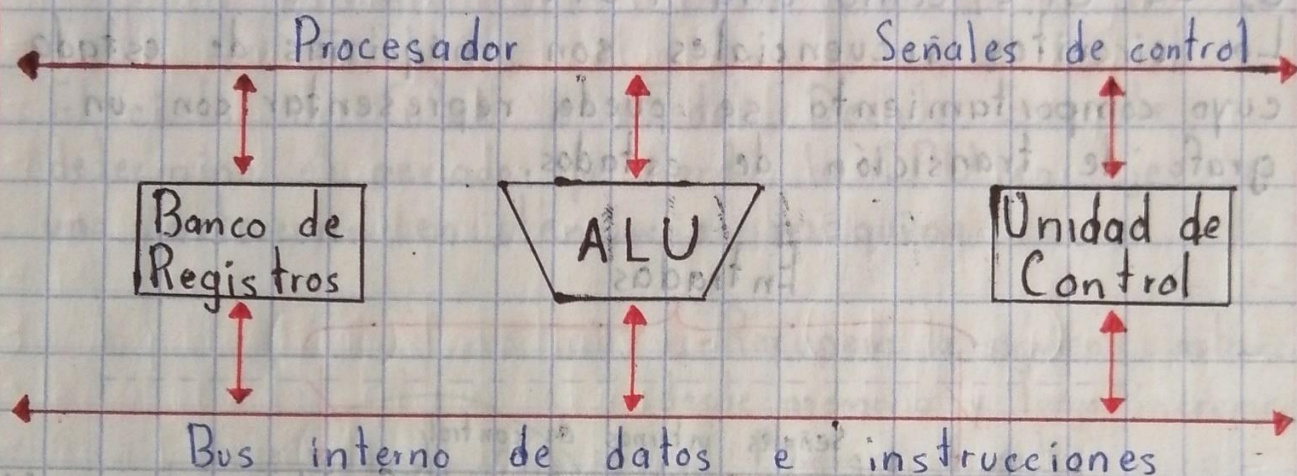
Los procesadores están diseñados y operan según una señal conocida como señal de reloj que es una señal en forma de onda cuadrada periódica por una determinada frecuencia.



5. Ejecución de las Instrucciones

Para ejecutar las instrucciones, todo procesador dispone de tres componentes principales:

- Conjunto de registros
- Unidad Aritmética Lógica (ALU)
- Unidad de Control



Dentro de la ejecución existe un ciclo el cual es la secuencia de operaciones para ejecutar las instrucciones. Se divide en cuatro fases principales:

1. Lectura de la instrucción. Las operaciones en esta fase son:

- a. Leer la instrucción
- b. Decodificar la instrucción
- c. Actualizar el contador del programa.

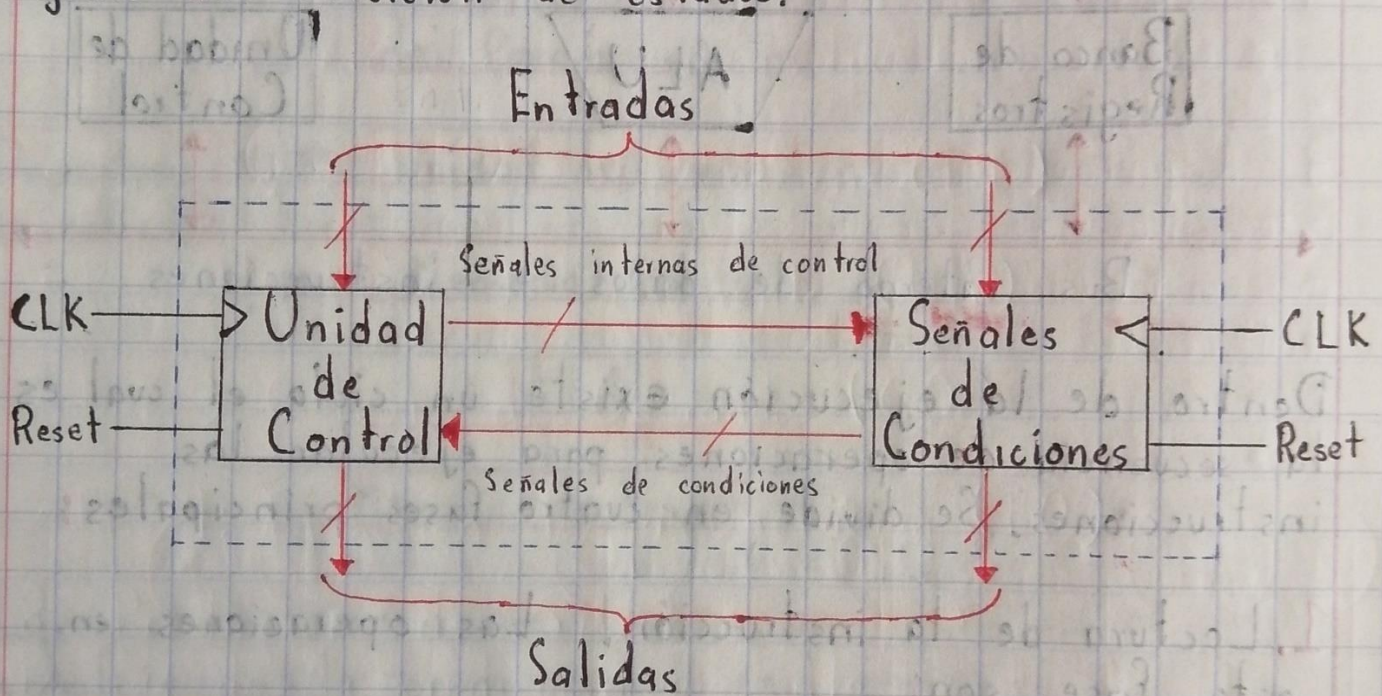
2. Lectura de los operandos fuente. Las operaciones a realizar dependen del modo de direccionamiento de los operandos

- a. Inmediato
- b. Directo a registro
- c. Entre otros

3. Ejecución de la instrucción y almacenamiento del operando de destino
4. Comprobación de interrupciones. Son el mecanismo para interrumpir un programa ejecutado por el procesador

6. La CPU como una FSM

Los circuitos secuenciales son máquinas de estado cuyo comportamiento se puede representar con un grafo de transición de estados.



La anterior figura muestra que la unidad de control es una máquina de estados que recibe información del exterior por señales de entrada.

Para diseñar un circuito secuencial que implemente una máquina de estados correspondiente a un controlador, se deben seguir los pasos descritos a continuación:

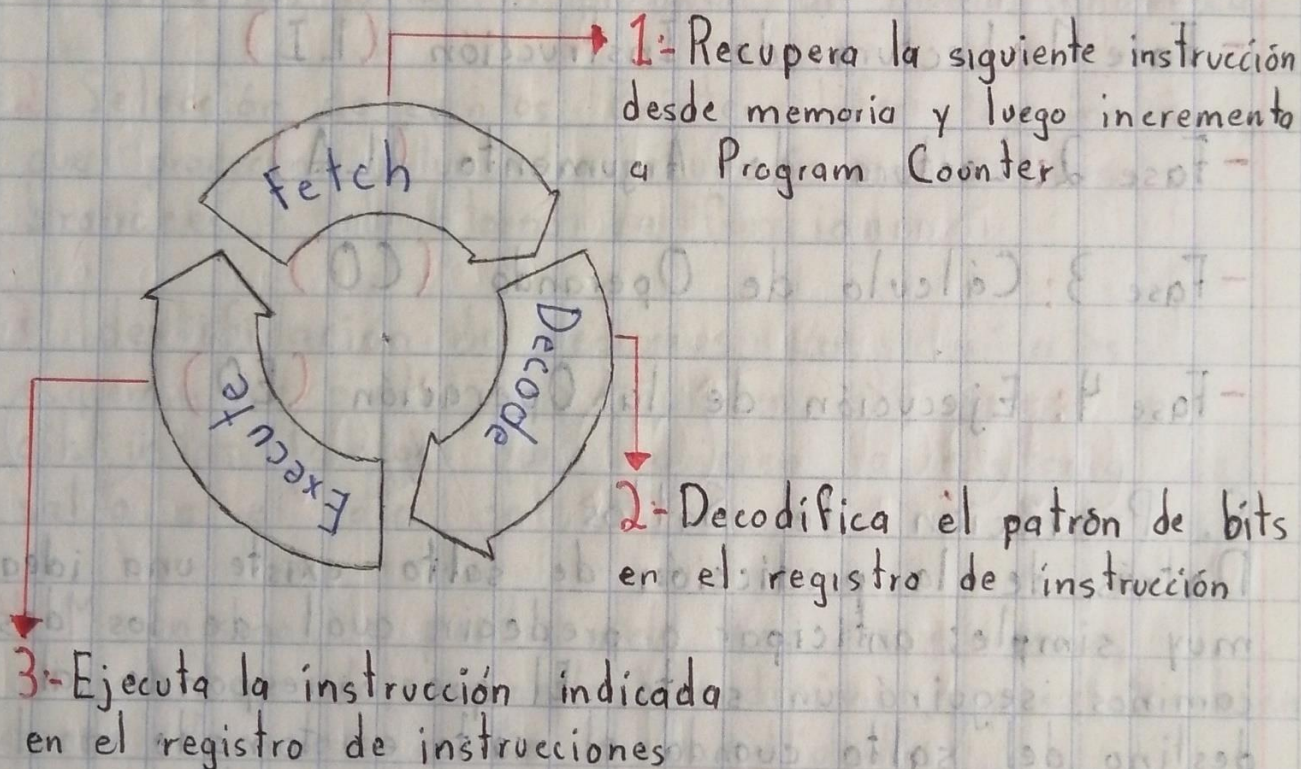
1. Decidir qué acciones se deben aplicar sobre el sistema que se controla.

2. Determinar las entradas del controlador

3. Diseñar la máquina de estados del controlador

Ciclo de Fetch

El ciclo de Fetch o Fetch Decode Execute determina el periodo que tarda la CPU en ejecutar una instrucción en lenguaje máquina



7. Pipeline

Para ir más rápido en el procesamiento de instrucciones se forma un pipeline. Una especie de tubería en la que pasa un flujo tratado en cada uno de sus segmentos

Instrucción	Etapa del pipeline						
	1	2	3	4	5	6	7
1	LI	LA	CO	EO			
2		LI	LA	CO	EO		
3			LI	LA	CO	EO	
4				LI	LA	CO	EO
5					LI	LA	CO
Periodo	1	2	3	4	5	6	7
	Latencia						

El ciclo de ejecución de una instrucción sería:

- Fase 1: Lectura de Instrucción (LI)
- Fase 2: Lectura de Argumento (LA)
- Fase 3: Cálculo de Operando (CO)
- Fase 4: Ejecución de la Operación (EO)

8. Predicción de Saltos

Detrás de la predicción de salto existe una idea muy simple: anticipar o predecir cuál de los dos caminos seguirá un salto. El que corresponde al destino del salto cuando el salto es tomado o el que continúa después de la instrucción de salto cuando éste no es tomado.

El comportamiento de los saltos puede predecirse estáticamente en tiempo de compilación y dinámicamente por medio del hardware en tiempo de ejecución.

Un ejemplo de planificación estática se da en el salto demorado cuando se deben seleccionar las instrucciones más convenientes para ser ubicadas en la ranura de demora.

Características de los predictores

Un predictor normalmente está acompañado de una fuente de eventos. En el caso específico de los predictores de salto tenemos:

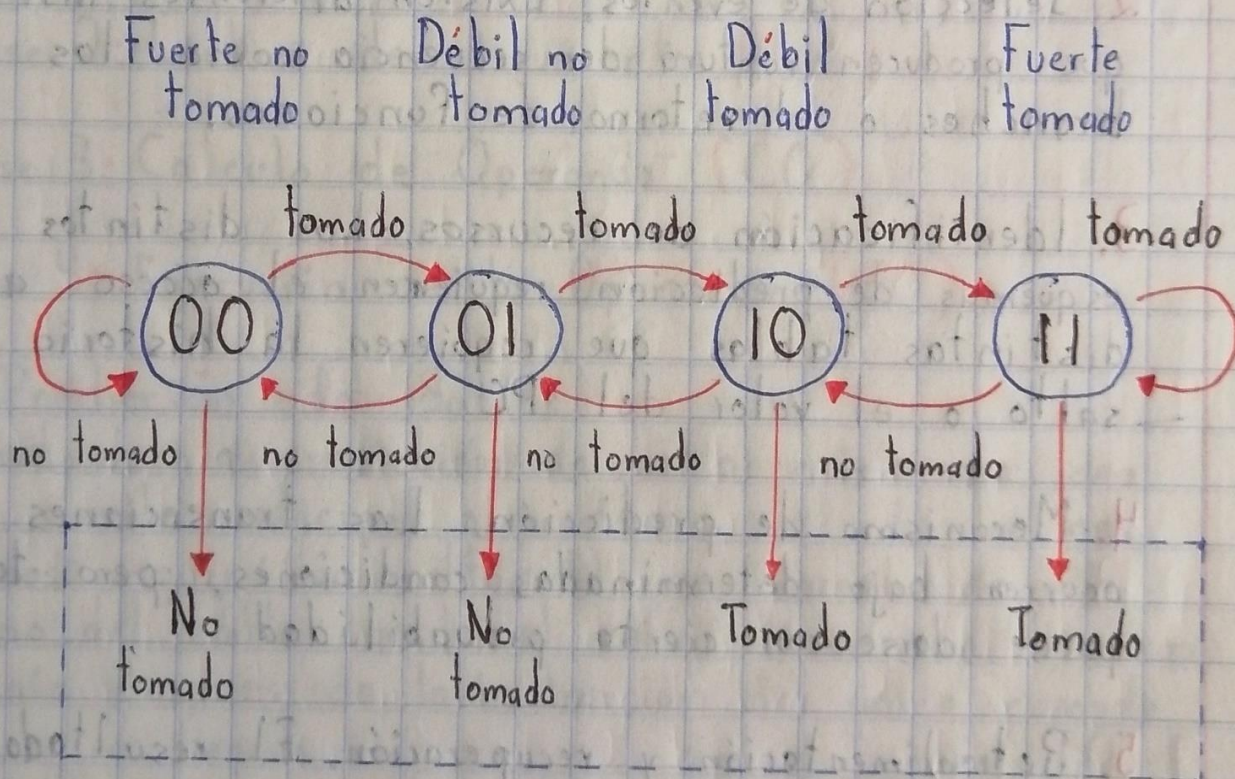
1. Origen de los eventos. Durante la ejecución del programa los saltos se producen de manera dinámica.
2. Selección de eventos. Existen diferentes instrucciones que producen ruptura de secuencia como los branches o el retorno de funciones.
3. Identificación de recursos. Los distintos esquemas de predicción requieren el acceso a distintas tablas que requieren la historia del salto o el valor del PC.
4. Mecanismo de predicción. Las transiciones ocurren bajo determinadas condiciones, pero también puede darse con cierta probabilidad.
5. Retroalimentación y recuperación. El resultado verdadero del salto se conoce algunos signos después de realizar la predicción. Si la predicción fue correcta, el mecanismo de retroalimentación debe reforzar la confianza del predictor.

Predictores locales

Esta clase de predictores se caracterizan porque usan solamente información relacionada a la instrucción de salto sobre la cual se hace la predicción

Predictores locales con contadores

Cuando el salto va a ser ejecutado, el valor del contador C es usado para hacer la predicción. Si C es mayor o igual a un límite predeterminado L , el salto se predice como tomado, en otro caso se predice como no tomado. Un valor típico para L es 2^{n-1}



9. Concepto de CISC y RISC

Dispositivos RISC

En la arquitectura computacional RISC (Reduced

Instruction Set Computer) es un tipo de microprocesador con las siguientes características:

- Instrucciones de tamaño fijo presentadas en un reducido número de formatos
- Sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos

Las máquinas RISC protagonizan la tendencia actual de construcción de microprocesadores:

"PowerPC", "Alpha", "MIPS", "ARM", "Spark", "Snap Dragon" son ejemplos de algunos de ellos.

Entre las ventajas de RISC tenemos:

1. La CPU trabaja más rápido al utilizar menos ciclos de reloj para ejecutar instrucciones
2. A diferencia de CISC, RISC conserva en memoria sus operaciones y operandos reduciendo la ejecución de nuevas operaciones
3. Cada instrucción puede ser ejecutada en un solo ciclo de CPU

Dispositivos CISC

En la arquitectura CISC (Complex Instruction Set Computer) es un modelo que tiene un conjunto de instrucciones que se caracterizan por ser muy amplio y permitir operaciones

complejas entre operandos.

Los CISC pertenecen a la primera corriente de construcción de procesadores antes del desarrollo de los RISC. Ejemplo de ellos son:

"Motorola 6800", "Zilog Z80" y toda la familia "Intel x86" usada en la mayoría de las computadoras personales actuales.

Entre las ventajas rescatables destacan:

1. Reduce la dificultad de crear compiladores

2. Permite reducir el costo total del sistema

3. Reduce los costos de creación del software

4. Mejora la compactación de código

5. Facilita la depuración de errores

10. Organización de datos en la memoria

Big Endian y Little Endian

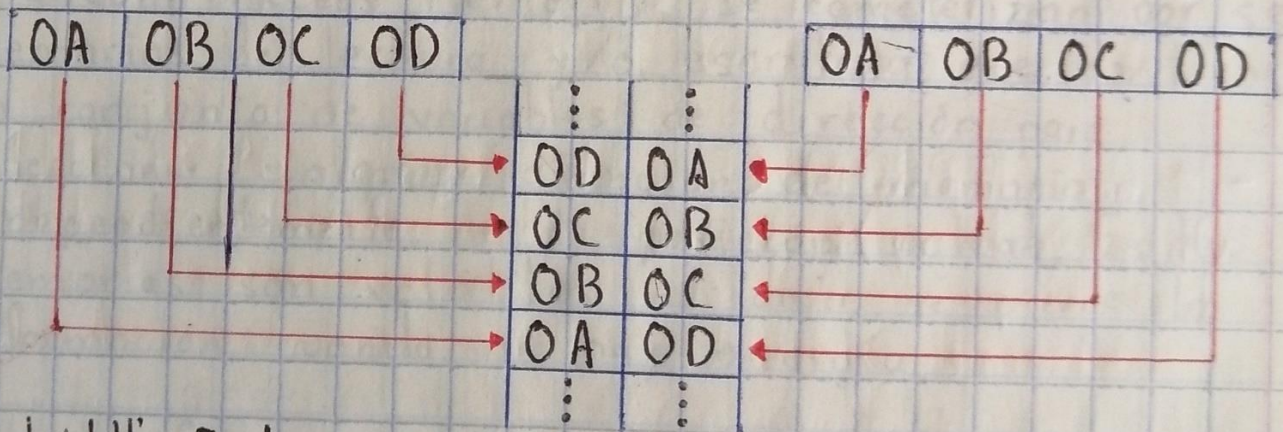
El término Endian se refiere a la forma en que los números binarios de bytes múltiples son guardados en la computadora. Por ejemplo los números binarios de 2 bytes pueden ser guardados en el formato de Big Endian (MSB LSB) o en el formato Little Endian donde MSB se refiere al bit más significativo y LSB se refiere al bit menos significativo.

Si requiere guardar el número 1 como un número binario de 2 bytes, puede escribirlo de la siguiente manera:

$0 \times 01 \rightarrow 00000000 00000001$

$0 \times 10 \rightarrow 00000001 00000000$

El formato en el que la computadora guarda la información puede usar ya sea Little Endian o Big Endian. Usted está leyendo un archivo guardado en formato Little Endian, para leer la información correcta debe leerlo y luego voltear los bytes para cambiarlos de Little Endian



Little Endian

Big Endian

Unidad 3:

Tipos de Memoria

Contenidos:

- Memoria aleatoria
- Memoria de sólo lectura
- Memoria de lectura y escritura
- Memoria secuencial
- Memoria caché

1. Memoria RAM

Las memorias de acceso aleatorio son conocidas como memorias RAM por sus siglas en inglés (Random Access Memory). Se caracterizan por ser memorias de lectura y/o escritura que contienen un conjunto de variables de dirección para seleccionar cualquier dirección de memoria independientemente de la posición actual. Estas memorias son volátiles, es decir, se pierde la información cuando no hay energía.

Se clasifican en dos categorías básicas:

- RAM estática: La SRAM (Static Random Access Memory) se compone de celdas conformadas por flip flops contruidos por transistores mosfet
- RAM dinámica: La DRAM (Dynamic Random Access Memory) se compone de celdas de memoria contruidas con capacitores, lo cual permite construir memorias de gran capacidad

2. Memoria ROM

Por sus siglas en inglés (Read Only Memory) se caracterizan por ser memorias de sólo lectura que contienen celdas de memoria no volátiles.

Se clasifican en:

- Memoria ROM de máscara: Se caracteriza porque la información contenida en su interior se almacena durante su fabricación y no puede ser alterado
- Memoria PROM (Programmable Read Only Memory): No se programa durante el proceso de fabricación, sino que la efectúa el usuario y se puede realizar una sola vez
- Memoria EPROM (Erasable Programmable Read Only Memory): Similar a la PROM con la diferencia que la información puede ser borrada y vuelta a escribir varias veces
- Memoria EEPROM (Electrical Erasable Programmable Read Only Memory): Pueden ser programadas y borradas varias veces por medio de la aplicación de un voltaje a una compuerta aislada "mosfet"
- Memoria Flash: Similar a la EEPROM, es decir que se puede programar y borrar eléctricamente. Se caracteriza por tener una capacidad muy elevada para almacenar información

3. Memorias de lectura y escritura

Se pueden establecer distintas clasificaciones atendiendo a distintos criterios. Algunas de estas son las siguientes:

Según el tiempo de permanencia

Se refiere al tiempo que permanece la información grabada en la memoria de las cuales podemos clasificar las siguientes:

- **No volátil o duradera:** Una vez escrito un punto de memoria su información no se borra hasta que se vuelve a escribir sobre él
- **Volátil:** La información desaparece si se deja de suministrar energía a la memoria
- **Con refresco:** Su información se degrada paulatinamente, llegando un momento en que no se puede leer correctamente. Para este tipo de memorias debe usarse un refresco periódicamente y reactivarse
- **Permanente o de sólo lectura:** Contienen siempre la misma información y no pueden borrarse

Según la forma de acceso

- **Acceso aleatorio:** De las cuales se desprenden las memorias activas y pasivas; éstas se refieren a que los tiempos de escritura y lectura son del

mismo orden. En cuanto a las memorias pasivas, los tiempos difieren siendo generalmente mucho mayor el de escritura.

- Acceso serie o secuencial: El tiempo de lectura y escritura depende del lugar de la memoria donde se quieran realizar las operaciones

Según la mínima información accesible

- De bit: Un ejemplo es un registro dentro de un microprocesador
- De palabra: Un ejemplo de éste son las memorias semiconductoras
- De bloque: Cuando se quiera leer un dato contenido en un bloque, primero debemos leer el bloque completo y luego una búsqueda del dato dentro del bloque

4. Memoria secuencial

Se caracterizan por tener tres principales clasificaciones:

- FIFO (First in, first out): La palabra que entró primero, será también la primera en salir
- LIFO (Last in, first out): La última palabra que entró será la primera en salir
- Memorias CCD (Charge Coupled Device): Todos los

bits realizan el mismo recorrido a través del lazo y a la misma frecuencia.

Este tipo de memorias opera con dispositivos capacitivos, que amplifican y regeneran la eficiencia en el proceso de transmisión

Unidad 4: El Microcontrolador 8051

Contenidos:

- Historia de los microprocesadores y microcontroladores
- Introducción al 8051
- Modos de direccionamiento
- Especificaciones eléctricas del 8051
- Juego de instrucciones
- Programas en línea recta
- Programas con lazos

1. Historia de los microprocesadores y microcontroladores
El dispositivo más reconocido para realizar cálculos es el ábaco, su origen se pierde en el tiempo, pero algunos estiman cerca de 3,000 años antes de Cristo.

A comienzo del Siglo XVI el sistema decimal desplazó al sistema romano en la realización de cálculos complejos. John Napier, matemático escocés, descubre los logaritmos y construye las primeras tablas de multiplicar a las que llamó Naiper Bones

Dispositivos mecánicos

Basados en logaritmos, surgen las primeras máquinas analógicas de cálculo, derivadas de prototipos contruidos por Edmund Gunter y William Oughtred. A este dispositivo se le llamó regla de cálculo.

El filósofo y matemático francés Blaise Pascal construye la primera máquina llamada Pascalina. Su funcionamiento se basaba en engranajes y ruedas con capacidad de sumar.

La pascalina se perfeccionó en una calculadora universal por Gottfried Leibniz, capaz de hacer cuatro operaciones aritméticas básicas. También la calculadora de doce dígitos por Philipp Matthäus y el telar de Jacquard controlado por tarjetas perforadas son otros avances significativos.

Era electromagnética

Con el surgimiento de la electricidad dio paso a la representación de dos estados estables: encendido y apagado, que en lógica digital sería 0 ó 1, junto con el trabajo realizado por George Boole.

El primer computador electromecánico llamado Mark 1 tenía 760,000 ruedas y 800 km de cable. Se basaba en una máquina analítica construida por Charles Babbage.

Era eléctrica

La invención del tubo de rayos catódicos de Karl Braun y el diodo de válvula de vacío por John A. Fleming precede en la invención al microprocesador.

ENIAC (Electronic Numerical Integrator And Computer)

Fue la primera computadora de propósito general desarrollada en la Universidad de Pennsylvania por John Presper Eckert y John William Mauchly

Era electrónica

La revolución del transistor comienza con su desarrollo e invención en los laboratorios de Bell Telephones para aparatos de radio, televisión, sonido, militar y la naciente tecnología de computadoras

El primer PC se construye en IBM lanzada al mercado en 1981. El modelo de IBM causó gran impacto por el concepto de compatibilidad. En su interior estaba el microprocesador 8088 de Intel

2. Introducción al 8088

El microcontrolador 8051 está basado en aquellos de 8 bits, 3 puertos de entrada y salida paralelos, un puerto de control, que a su vez contiene un puerto serie, dos entradas para timer de 16 bits, dos entradas para interrupciones externas, las señales RB y WR para la toma o almacenamiento de datos externos en RAM, la señal PSEN para lectura de instrucciones almacenadas en la EPROM externa.

Gracias a estas tres señales el microcontrolador 8051 puede direccionar 64 kilobytes de datos separados, es decir, 128 kilobytes. Además cuenta con 128 bytes de memoria RAM interna.

Puerto
1

PI.0 - 1
PI.1 - 2
PI.2 - 3
PI.3 - 4
PI.4 - 5
MOST - PI.5 - 6
MISO - PI.6 - 7
SCK - PI.7 - 8
RST - 9

Puerto
3

RXD - P3.0 - 10
TXD - P3.1 - 11
INT0 - P3.2 - 12
INT1 - P3.3 - 13
T0 - P3.4 - 14
T1 - P3.5 - 15
 $\overline{WR}$ - P3.6 - 16
 $\overline{RD}$ - P3.7 - 17
XTAL2 - 18
XTAL1 - 19
GND - 20

40 - VCC

39 - P0.0 - AD0

38 - P0.1 - AD1

37 - P0.2 - AD2

36 - P0.3 - AD3

35 - P0.4 - AD4

34 - P0.5 - AD5

33 - P0.6 - AD6

32 - P0.7 - AD7

31 - $\overline{EA}$ / VPP

30 - ALE / $\overline{PROG}$

29 - $\overline{PSEN}$

28 - P2.7 - A15

27 - P2.6 - A14

26 - P2.5 - A13

25 - P2.4 - A12

24 - P2.3 - A11

23 - P2.2 - A10

22 - P2.1 - A9

21 - P2.0 - A8

Puerto
0

Puerto
2

3. Modos de direccionamiento

Cuando las instrucciones operan sobre los datos, surge la pregunta: ¿En dónde están los datos? cuya respuesta encontraremos en los modos de direccionamiento del 8051.

Existen varios modos de direccionamiento posibles y varias respuestas posibles a dicha pregunta tales como "en el byte 2 de la instrucción", "en el registro R4", "en la dirección directa 35H" o tal vez "en la memoria externa para datos en la dirección contenida en el apuntador de datos".

Existen ocho modos disponibles:

Direccionamiento inmediato

Cuando el operando de origen es una constante en vez de una variable, entonces la constante puede incorporarse como un byte de dato "inmediato" a la instrucción. En lenguaje ensamblador, los operandos inmediatos están precedidos del símbolo "#". Por ejemplo la instrucción:

```
MOV A, #12
```

Todas las instrucciones que utilizan direccionamiento inmediato, a excepción de una, usan una constante de 8 bits para el dato inmediato. Se requiere de una constante de 16 bits cuando se inicializa el apuntador de datos. Por ejemplo:

MOV DPTR, #8000H

es una instrucción de 3 bytes que carga la constante de 16 bits 8000H al apuntador de datos. La forma general de esta instrucción es:

MOV A, #datos

Direccionamiento por registro

El programador del 8051 tiene acceso a ocho "registros funcionales", enumerados del R0 al R7.

Podemos combinar un código de función y la dirección de un operando para formar una instrucción corta (de 1 byte); por ejemplo:

MOV A, R7

La forma general de las instrucciones para transferir bytes es:

MOV byte_destino, byte_origen

Direccionamiento directo

El direccionamiento directo puede acceder a cualquier variable incorporada en el chip o a cualquier registro del hardware.

Se selecciona uno de dos espacios de memoria incorporados en el chip, dependiendo del bit de mayor orden de la dirección directa.

Así, por ejemplo la instrucción:

`MOV P1, A`

transfiere el contenido del acumulador al puerto 1. Ahora, la forma general de esta instrucción es:

`MOV directo, #dato`

Direccionamiento indirecto

En el 8051, los registros R0 y R1 pueden operar como registros "apuntadores". Este direccionamiento en lenguaje ensamblador del 8051 se representa usando el signo de arroba (@) como prefijo para R0 o R1. Por ejemplo, si el registro R1 contiene el valor 40H y la dirección de memoria interna 40H contiene el valor 55H, la instrucción:

`MOV A, @R1`

transfiere el valor 55H al acumulador. Entonces, la forma general de esta instrucción es:

`MOV A, @Ri`

Direccionamiento indexado

Éste utiliza un registro base (ya sea el contador de un programa o el apuntador de datos) y un desplazamiento (el acumulador) para formar la dirección efectiva para una instrucción JMP o MOVC.

La forma general de esta instrucción es:

`MOVC A, @A + DPTR`

Direccionamiento relativo

Éste sólo se utiliza con ciertas instrucciones de salto. El rango de salto es de -128 a $+127$ ubicaciones. El desplazamiento relativo se añade a la instrucción en forma de un byte adicional.

Por ejemplo, si la etiqueta `AQUI` representa una instrucción en la ubicación `1040H` y la ubicación con la instrucción:

`SJMP AQUI`

está ubicada en las direcciones `1000H` y `1001H` de la memoria, el ensamblador asignará un desplazamiento relativo con el valor `3EH` como el byte 2 de la instrucción ($1002H + 3EH = 1040H$). La forma general de la instrucción es:

`SJMP direccion_relativa`

Direccionamiento absoluto

Las instrucciones `ACALL` y `AJMP` son las únicas que utilizan el direccionamiento absoluto. Estas instrucciones de 2 bytes proporcionan los 11 bits menos significativos de la dirección de destino en el código de operación (`A10-A8`) y el byte 2 de la instrucción (`A7-A0`) que permite la bifurcación dentro

de la página de 2K actual de la memoria para código.

Direccionamiento largo

Las instrucciones LCALL y LJMP son las únicas que utilizan el direccionamiento largo. Estas instrucciones de 3 bytes incluyen una dirección de destino de 16 bits, representada por los bytes 2 y 3 de la instrucción. La ventaja es que puede utilizar todo el espacio de código de 64K, pero la desventaja es que las instrucciones son de 3 bytes y dependen de su posición.

4. Retardo

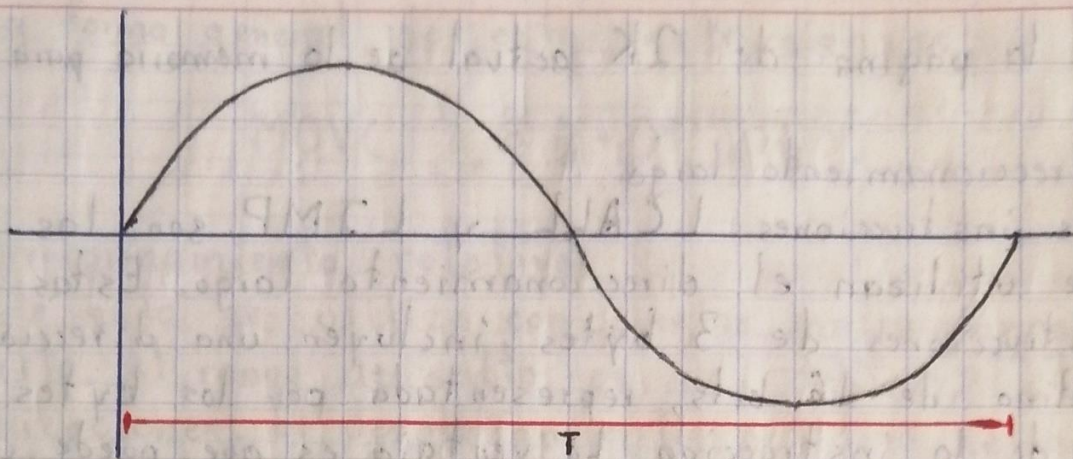
En ensamblador, la elaboración de un retardo no se maneja con alguna función, por lo que uno mismo debe crearlo. Para ello se deben considerar varios factores.

El factor más importante es el factor velocidad. Se debe conocer a qué velocidad trabaja el micro. Hay algunos que trabajan a 8 MHz, otros que trabajan a 12 MHz y otros a 16 MHz. El 8051 trabaja a 12 MHz.

La representación de 12 MHz es una frecuencia. Una frecuencia es igual a la inversa del periodo:

$$f = \frac{1}{T}$$

Recordemos que el periodo es toda una transición senoidal



Por lo tanto, una frecuencia de 12 MHz implica un periodo de 8.333×10^{-8} segundos. También es importante definir un ciclo máquina. Un ciclo máquina es el tiempo que le toma al microprocesador acabar una instrucción. Para el caso del 8051, un ciclo máquina le lleva 1×10^{-6} segundos. Así, si se quiere un retardo de un segundo, deberán consumirse 1,000,000 de ciclos máquina.

5. PSW (Palabra de estado del programa o Program Status Word)

La PSW ubicada en la dirección D0H, contiene los bits de estado:

Bandera de acarreo

La bandera de acarreo (C o CY) es de doble propósito. Se utiliza en la manera tradicional para realizar operaciones aritméticas: se vuelve 1 si hay un acarreo del bit 7 durante una suma o si hay un préstamo hacia el bit 7 durante una resta.

Bandera auxiliar de acarreo

La bandera auxiliar de acarreo (AC) se establece en 1 cuando se realiza una suma de valores decimales codificados en binario (BCD), siempre y cuando se haya generado un acarreo del bit 3 al bit 4 o si el resultado en el nibble inferior se encuentra en el rango 0AH-0FH.

Bandera 0

La bandera 0 (FO) es un bit de bandera de propósito general disponible para aplicaciones de usuario.

Bits de selección del banco de registros

Éstos (RS0 y RS1) determinan el banco de registros que está activo. Estos bits se restablecen en 0 después de un reinicio del sistema y el software puede cambiarlos de acuerdo a sus necesidades.

Bandera de desbordamiento

La bandera de desbordamiento (OV) cambia a 1 después de una operación de suma y resta si hubo un desbordamiento aritmético. El software puede examinar esta bandera para determinar si el resultado se encuentra dentro del rango apropiado.

Bit de paridad

El bit de paridad (P) cambia a 1 o a 0 de manera automática en cada ciclo de máquina para establecer la paridad par con el acumulador. La suma del número de bits en 1 en el acumulador más el bit P siempre es un resultado par.

6. Tipos de instrucciones

Las instrucciones del 8051 se dividen en cinco grupos funcionales:

- Aritméticas
- Lógicas
- De transferencia de datos
- Booleanas
- De bifurcación de programa

Instrucciones aritméticas

La instrucción ADD A puede escribirse de distintas maneras, ya que hay cuatro posibles modos de direccionamiento:

- ADD A, 7PH (direccionamiento directo)
- ADD A, @R0 (direccionamiento indirecto)
- ADD A, R7 (direccionamiento por registro)
- ADD A, #35H (direccionamiento inmediato)

La instrucción MUL AB multiplica el acumulador por el dato localizado en el registro B y transfiere el producto de 16 bits al registro B (byte superior) y al acumulador (byte inferior), ambos están concatenados.

La instrucción `DIV AB` divide el acumulador entre el dato localizado en el registro B, dejando el cociente de 8 bits en el acumulador y el residuo de 8 bits en el registro B.

Nota: Todas las instrucciones aritméticas se ejecutan en un ciclo máquina excepto `INC DPTR` (dos ciclos máquina) y `MUL AB` y `DIV AB` (cuatro ciclos máquina).

Instrucciones lógicas

Las instrucciones lógicas del 8051 realizan operaciones booleanas (`AND`, `OR`, `OR` exclusivo y `NOT`) sobre bytes de datos, bit por bit. Si el acumulador contiene el valor `00110101B`, la siguiente instrucción lógica `AND`:

`ANL A, #01010011B`

deja el valor `00010001B` en el acumulador.

La instrucción lógica `AND` puede tomar diferentes formas debido a los modos de direccionamiento:

- `ANL A, 55H` (direccionamiento directo)
- `ANL A, @R0` (direccionamiento indirecto)
- `ANL A, R6` (direccionamiento por registro)
- `ANL A, #33H` (direccionamiento inmediato)

Todas las instrucciones lógicas que utilizan el acumulador como uno de los operandos se ejecutan en un ciclo de máquina. Las otras instrucciones necesitan de dos ciclos de máquina.

Las operaciones lógicas pueden realizarse en cualquier byte en el espacio de memoria interna para datos, sin tener que acceder al acumulador.

La instrucción:

XRL directo, #datos

ofrece una manera fácil y rápida de invertir los bits de los puertos.

Las instrucciones de rotación (RLA y RRA) desplazan una posición a la izquierda o a la derecha respectivamente, todos los bits del acumulador. Las variaciones de estas instrucciones, RLC A y RRC A son rotaciones de 9 bits que utilizan el acumulador y la bandera de acarreo incluidos en la PSW.

La instrucción SWAP A intercambia los nibbles superior e inferior dentro del acumulador. Esta operación es muy útil para manipular números en BCD.

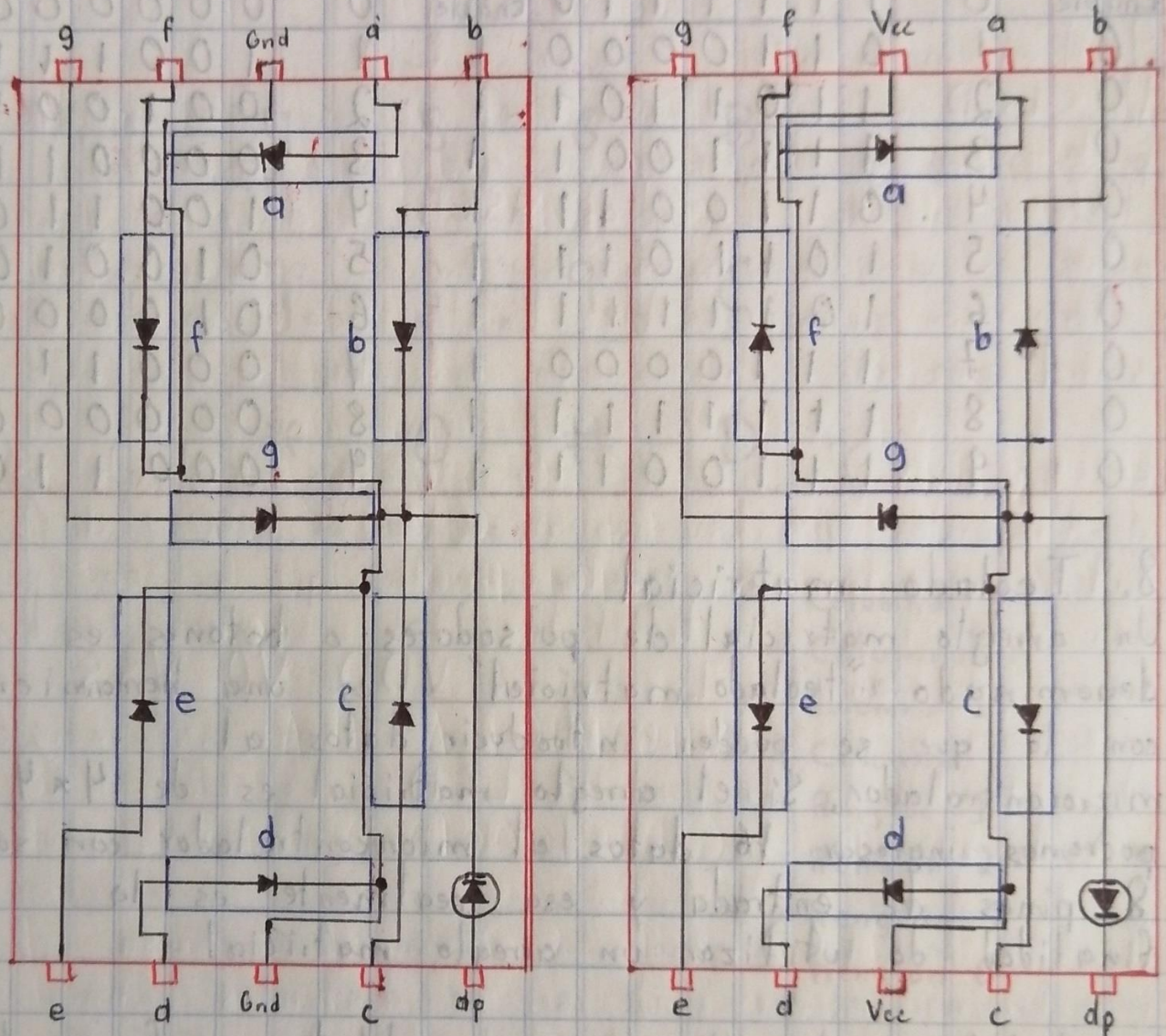
7. Display de 7 segmentos

El display de 7 segmentos es una forma de representar números en equipos electrónicos. Está compuesto de siete segmentos que se pueden encender o apagar individualmente

y cada segmento tiene la forma de una pequeña línea.

Cátodo común

Ánodo común



		Cátodo Común									Ánodo Común								
		Número	A	B	C	D	E	F	G			Número	A	B	C	D	E	F	G
Enable	0	0	1	1	1	1	1	1	0	Enable	0	0	0	0	0	0	0	0	1
0	1	0	1	1	0	0	0	0	0	1	1	1	0	0	1	1	1	1	1
0	2	1	1	0	1	1	0	1	1	1	2	0	0	1	0	0	1	0	0
0	3	1	1	1	1	0	0	1	1	1	3	0	0	0	0	1	1	0	0
0	4	0	1	1	0	0	1	1	1	1	4	1	0	0	1	1	0	0	0
0	5	1	0	1	1	0	1	1	1	1	5	0	1	0	0	1	0	0	0
0	6	1	0	1	1	1	1	1	1	1	6	0	1	0	0	0	0	0	0
0	7	1	1	1	0	0	0	0	0	1	7	0	0	0	1	1	1	1	1
0	8	1	1	1	1	1	1	1	1	1	8	0	0	0	0	0	0	0	0
0	9	1	1	1	0	0	1	1	1	1	9	0	0	0	1	1	0	0	0

8. Teclado matricial

Un arreglo matricial de pulsadores o botones es denominado "teclado matricial" y es una herramienta con la que se pueden introducir datos al microcontrolador. Si el arreglo matricial es de 4×4 podremos ingresar 16 datos al microcontrolador con sólo 8 pines de entrada y esa realmente es la finalidad de utilizar un arreglo matricial.

Los teclados matriciales son ensamblados en forma de matriz. Cuando no se ha oprimido ninguna tecla (todas las teclas abiertas) no hay conexión entre renglones y columnas. Cuando se oprime una tecla se hace una conexión entre la columna y el renglón de la tecla.

1	2	3	A
4	5	6	B
7	8	9	C
*	0	#	D

Columna 3

Columna 2

Columna 1

Columna 0

Renglón 3

Renglón 2

Renglón 1

Renglón 0

La forma de ingresar datos al microcontrolador es:

- Definir las filas y columnas de la matriz y sus respectivos pines asignados a ella
- Realizar un barrido ya sea por fila o por columna y definir si el barrido será con estado alto o bajo

- Al realizar el barrido se debe volver a leer el puerto o pines asignados para verificar que una tecla sea presionada o no

9. Funcionamiento de una LCD

Un LCD es un dispositivo de visualización que puede desplegar información gráfica y en modo de texto cubriendo el código ASCII, es decir los caracteres numéricos y alfanuméricos necesarios para mostrar información comprensible. Existen tanto con comunicación serial como con comunicación paralela.

Existen LCD's de distintos tamaños, pudiendo ser distinguidos por su variación en el número de filas y columnas, existen aquellos que poseen sólo una fila hasta cuatro filas; las columnas también varían de tamaño. Los pines VDD, VO, VSS son conectados a un potenciómetro con el cual se controla el contraste de la pantalla LCD, ajustando manualmente la claridad de la información en pantalla.

También los LCD poseen en algunos casos una retroiluminación que puede ser de distintos colores como verde, blanco, amarillo, azul, etc. La alimentación para la retroiluminación es a través de los pines denominados A y K que representan el ánodo y el cátodo. Es recomendable alimentar estos pines con una resistencia de protección para no quemar la retroiluminación del LCD si éste lo posee.

Los LCD's tienen dos modos de trabajo, uno donde se utilizan los ocho pines del bus de datos para el envío y recepción de datos, información y comandos y otro donde sólo se utilizan cuatro. En el modo de cuatro bits se ahorra la cantidad de pines necesarios para controlar el LCD lo cual es conveniente en caso de que se requiera utilizar microcontroladores más pequeños que no tengan muchos recursos pero que sean más económicos y suficientes para el propósito del diseñador.

PIN	Símbolo	Nivel	Función
1	VSS	-	GND (0V)
2	VDD	-	Fuente de Voltaje a Logic (+5V)
3	VO	-	Fuente de poder al LCD
4	RS	H/L	H: Datos L: Código de instrucción
5	R/W	H/L	H: Leer L: Escribir
6	E	H/L	Señal Enable
7	DB0	H/L	Línea de datos del bus
8	DB1	H/L	
9	DB2	H/L	
10	DB3	H/L	
11	DB4	H/L	
12	DB5	H/L	
13	DB6	H/L	
14	DB7	H/L	
15	LEDA	H/L	Backlight Power (+5V)
16	LEDK	-	Backlight Power (0V)

Unidad 5:

Sistema Mínimo y Periféricos

Contenidos:

- Elementos básicos de un sistema mínimo
 - Circuito de Reset
 - Circuito de Reloj
 - Puertos paralelos
- Instalación de memoria externa en un sistema mínimo
- Interrupciones
 - Mecanismo de servicio
 - Vector de interrupciones
 - Externas
- Timers
- Puerto Serie
- Aplicación del procesador a un proyecto real

1. Interrupciones

En microprocesadores se define como una interrupción que se realiza al proceso que se está llevando a cabo en ese momento en el procesador, de modo que éste debe dejar la labor que estaba ejecutando para atender la interrupción solicitante. Una vez atendida, puede retomar el proceso donde lo dejó. Es como un subprograma pero puede llamarse vía interrupción por hardware.

Interrupciones del microcontrolador

Dependiendo del modelo se tienen distintas interrupciones. Por ejemplo, el AT89C51 de la familia del 8051 posee los siguientes recursos que solicitan una interrupción:

- Interrupción externa 0
- Timer 0
- Interrupción externa 1
- Timer 1
- Puerto serie Rx y Tx

Adicionalmente posee un Timer 2, más memoria de programa y de datos y un vector de interrupciones.

Interrupt Enable Register (IE)

Si ocurren dos o más interrupciones de manera simultánea, o si una interrupción ocurre al mismo tiempo que se le está dando servicio a otra, existe la secuencia de sondeo y un esquema de prioridades para planificar interrupciones.

La secuencia de sondeo es fija pero la prioridad de interrupción es programable. Cada una de las fuentes de interrupción se habilita o deshabilita en forma individual mediante el registro de función especial IE (habilitador de interrupción) que es direccionable por bit y está ubicado en la dirección 0A8H.

Prioridad de las interrupciones

Cada fuente de interrupción se programa de manera individual con uno de dos niveles de prioridad mediante el registro de función especial

direccionable por bit IP (prioridad de interrupción)
en la dirección 0B8H

Interrupt Priority Register (IP)

El registro IP se borra después de una reinicialización del sistema para que todas las interrupciones empiecen en el nivel de prioridad más bajo de manera predeterminada. Si una ISR de baja prioridad se está ejecutando cuando ocurre una interrupción de alta prioridad, esta ISR se interrumpe. Una ISR de alta prioridad no puede ser interrumpida.

Resumen de registro IP (prioridad de interrupción)			
Bit	Símbolo	Dirección de bit	Descripción (1=alto nivel, 0=bajo nivel)
IP.7	—	—	Indefinido
IP.6	—	—	Indefinido
IP.5	PT2	0BDH	IP del temporizador 2 (8052)
IP.4	PS	0BCH	IP del puerto serial
IP.3	PT1	0BBH	IP del temporizador 1
IP.2	PX1	0BAH	IP de interrupción externa 1
IP.1	PT0	0B9H	IP del temporizador 0
IP.0	PX0	0B8H	IP de interrupción externa 0

Secuencia de sondeo

Si ocurren dos interrupciones de la misma prioridad simultáneamente, una secuencia de sondeo fija determina a cuál se le dará servicio primero. La secuencia de sondeo es: externa 0, temporizador 0, externa 1, temporizador 1, puerto serial, temporizador 2.

Procesamiento de interrupciones

El programa principal se interrumpe cuando ocurre una interrupción y ésta es aceptada por la CPU. Esto causa las siguientes reacciones:

- La instrucción actual completa su instrucción
- El contador de programa (PC) se almacena en la fila
- El estado de la interrupción actual se almacena internamente
- Las interrupciones se bloquean al nivel de interrupción
- El PC se carga con la dirección de vector de la rutina de servicio de interrupción (ISR)
- La ISR se ejecuta

Vectores de interrupción

Cuando se acepta una interrupción, el valor cargado en el contador de programa es conocido como vector de interrupción. Este valor es la dirección del inicio de la ISR para la fuente que causó la interrupción.

Cuando "vectorizamos una interrupción", el hardware borra de manera automática la bandera que causó la interrupción. Las excepciones a esto son RI y TI para las interrupciones del puerto serial y TF2 y EXF2 para el temporizador 2. No es práctico que la CPU limpie la bandera de instrucción debido a que

existen dos fuentes posibles para cada una de estas interrupciones

Vectores de Interrupción		
Interrupción	Bandera	Dirección de vector
Reinicio	RST	0000H
Externa 0	IE0	0003H
Temporizador 0	TF0	000BH
Externa 1	IE1	0013H
Temporizador 1	TF1	001BH
Puerto serial	RI o TI	0023H
Temporizador 2	TF2 o EXF2	002BH

2. Timers

Un temporizador consta de una serie de flip flops de división entre dos, que reciben su fuente de reloj como entrada. Cada etapa sucesiva divide entre 2^n . Podemos considerar al valor binario de las salidas de los flip flops del temporizador como la "cuenta" del número de pulsos de reloj desde que el temporizador inició.

Los temporizadores se utilizan en casi todas las aplicaciones orientadas al control. El 8051 tiene dos temporizadores de 16 bits, cada uno con cuatro modos de operación. Un tercer temporizador de 16 bits con tres modos de operación se le añadió al 8052. Los temporizadores se utilizan para la temporización de intervalos, el conteo de eventos o la generación de la tasa de transmisión y recepción en baudios para el puerto serial incorporado.

Registros con funciones especiales de un temporizador			
SFR	Propósito	Dirección	Direccionable por bit
TCON	Control	88H	Si
TMOD	Modo	89H	No
TLO	Byte inferior de temp. 0	8AH	No
TL1	Byte inferior de temp. 1	8BH	No
TH0	Byte superior de temp. 0	8CH	No
TH1	Byte superior de temp. 1	8DH	No
T2CON*	Control de temp. 2	8EH	Si
RCAP2L*	Captura byte inferior de t2	8FH	No
RCAP2H*	Captura byte superior de t2	90H	No
TL2*	Byte inferior de temp. 2	91H	No
TH2*	Byte superior de temp. 2	92H	No

Registro de modo del temporizador (TMOD)

El registro TMOD contiene dos grupos de cuatro bits, que establecen el modo de operación para los temporizadores 0 y 1. Los bits de TMOD no son direccionables individualmente ni requieren serlo. Por lo general, el registro se carga una sola vez mediante el software al inicio de un programa para inicializar el modo del temporizador. Después de esto, puede detenerse, iniciarse, etc. por acceso a otros SFR del temporizador.

Registro de control del temporizador (TCON)

El registro TCON contiene los bits de estado y de control para los temporizadores 0 y 1. Los cuatro bits superiores en el TCON (TCON.4 - TCON.7) se utilizan para encender o apagar los

temporizadores (TR0, TR1) o para indicar el desbordamiento del temporizador (TF0, TF1). Los cuatro bits inferiores en TCON (TCON.0 - TCON.3) no tienen nada que ver con los temporizadores. Estos se utilizan para detectar e iniciar interrupciones externas.

Modos del temporizador y la bandera de desbordamiento
Utilizamos la notación "x" para referirnos a los temporizadores 0 y 1 indistintamente.

Modo de temporizador de 13 bits (Modo 0)

El modo 0 es un modo de temporizador de 13 bits que provee compatibilidad con el predecesor del 8051: el 8048. Éste generalmente no se utiliza en los nuevos modelos.

El byte superior del temporizador (THx) está dispuesto en cascado con los cinco bits menos significativos del byte inferior del temporizador (TLx) para formar un temporizador de 13 bits. Los tres bits superiores del TLx no son utilizados.

Modo de temporizador de 16 bits (Modo 1)

Es igual al modo 0, excepto que el temporizador opera como temporizador de 16 bits. La bandera de desbordamiento es el bit TFX en TCON, el cual se lee o escribe mediante el software.

El bit más significativo del valor en los registros del temporizador es el bit 7 de THx y el bit

menos significativo es el bit 0 de TLx. El LSB se dispara a una frecuencia de reloj de entrada dividida entre 2, mientras que el MSB se dispara a una frecuencia de reloj de entrada dividida entre 65,536.

Modo de temporizador de 8 bits (Modo 2)

Es un modo de autotrecarga de 8 bits. El byte inferior TLx opera como temporizador de 8 bits mientras que THx almacena un valor de recarga.

Este modo es muy útil ya que el desbordamiento del temporizador ocurre a intervalos periódicos específicos una vez que TMOD y THx se han inicializado.

Modo de temporizador dividido (Modo 3)

Es un modo de temporizador dividido y es distinto para cada temporizador.

El temporizador 0 en el modo 3 se divide en dos temporizadores de 8 bits. TLO y THO actúan como temporizadores separados, en donde los desbordamientos establecen en 1 los bits TFO y TF1.

En el modo 3, el temporizador 1 está detenido, pero puede iniciarse al cambiarlo a uno de los otros modos con la limitación de que la bandera de desbordamiento TF1 no se ve afectada por estar conectada a THO.

Fuentes de reloj

Existen dos fuentes posibles de reloj, las cuales se seleccionan la escritura al bit de contador/temporizador ($C/\bar{T}$) en TMOD cuando se inicializa el temporizador. Una fuente de reloj se utiliza para la temporización de intervalos y la otra para conteo de eventos.

Temporización de intervalos

Si $C/\bar{T} = 0$ se selecciona la operación continua del temporizador y éste recibe la señal de reloj del oscilador incorporado en el chip. Se agrega una etapa de división entre 12 para reducir la frecuencia del reloj a un valor razonable para la mayoría de las aplicaciones.

Inicio, detención y control de los timers

El método más simple para iniciar y detener los timers es mediante el bit de control de arranque, TRx en TCON. TRx se pone en 0 después de una reinicialización del sistema; por lo tanto, los temporizadores estarán detenidos.

El temporizador 0 se iniciará mediante:

`SETB TRO`

y se detiene con:

`CLR TRO`

Ensamblador convertirá de TRO a la dirección de bit correcta.

3. Puerto Serie

El 8051 incluye un puerto serial incorporado al chip, el cual puede operar en diversos modos sobre un extenso rango de frecuencias. La función esencial del puerto serial es realizar una conversión en paralelo a serial para los datos de salida y una conversión de serial a paralelo para los datos de entrada. El acceso al puerto serial mediante el hardware se realiza a través de las terminales TXD y RXD. Estas terminales constituyen las funciones alternas para dos bits del puerto 3, P3.1 en la terminal 11 (TXD) y P3.0 en la terminal 10 (RXD).

El puerto serial cuenta con una operación full duplex (transmisión y recepción simultáneas) y con búfer de recepción, lo cual permite recibir un carácter que se almacena en un búfer mientras se recibe un segundo carácter.

Comunicación serial

Este tipo de comunicación involucra la transmisión de bits de datos a través de una sola línea de comunicación. Los datos se transmiten bit por bit, ya sea en formato sincrónico o asincrónico.

La comunicación serial sincrónica transmite un bloque completo de caracteres sincronizados mediante un reloj de referencia, mientras que la comunicación serial asincrónica transmite de manera aleatoria un carácter en cualquier

momento, independientemente de cualquier reloj.

Registro del búffer del puerto serial (SBUF)

El registro de búffer del puerto serial (SBUF) localizado en la dirección 99H constituye, en realidad, dos memorias intermedias.

Registro de control del puerto serial (SCON)

El registro de control del puerto serial (SCON) ubicado en la dirección 98H es un registro direccionable por bit que contiene los bits de estado y los bits de control.

Los bits de estado indican el fin de una transmisión o recepción de un carácter y se prueban en el software o se programan para producir una interrupción.

Modos del puerto serial

SM0	SM1	Modo	Descripción	Velocidad en baudios
0	0	0	Registro	Fija (frecuencia del oscilador $\div 12$)
0	1	1	UART de 8 bits	Variable (establecida por timer)
1	0	2	UART de 9 bits	Fija (frecuencia del oscilador $\div 12 \times 64$)
1	1	3	UART de 9 bits	Variable (establecida por timer)

Modos de operación

El puerto serial del 8051 tiene cuatro modos de operación, los cuales se seleccionan en el SCON mediante la escritura de unos y ceros a los bits SM0 y SM1. Tres de estos modos habilitan comunicaciones asíncronas, en donde cada carácter

recibido o transmitido está emmarcado por un bit de inicio y un bit de detención.

Registro de desplazamiento de 8 bits (Modo 0)

El modo 0, el cual se selecciona al escribir ceros en los bits SM1 y SM0 del SCON, configura el puerto serial en el modo cero. Los datos seriales entran y salen a través de RXD y TXD envía la salida del reloj de desplazamiento.

La velocidad en baudios está fija a un doceavo de la frecuencia del oscilador incorporado al chip. En este modo se transmiten o reciben ocho bits, con el menos significativo primero.

UART de 8 bits con velocidad en baudios variable (Modo 1)

El modo 1 del puerto serial del 8051 le permite operar como un UART de 8 bits con velocidad en baudios variable. Un receptor/transmisor asincrónico universal (UART, por sus siglas en inglés) es un dispositivo que recibe y transmite datos seriales y donde cada carácter va precedido por un bit de inicio (bajo) y seguido por un bit de detención.

A veces se añade un bit de paridad entre el último bit de los datos y el bit de detención.

La operación esencial de un UART es la conversión paralelo a serial de los datos de salida y la conversión serial a paralelo de los datos de entrada.

UART de 9 bits con velocidad en baudios fija (Modo 2)

El puerto serial opera en modo 2 cuando $SM1=1$ y $SM0=0$. Se transmiten o reciben once bits: un bit de inicio, ocho bits de datos, un noveno bit de datos programable y un bit de detención.

UART de 9 bits con velocidad en baudios variable (Modo 3)

El modo 3, un UART de 9 bits con velocidad en baudios variable es igual que el modo 2, excepto que la velocidad en baudios es programable y la proporciona el temporizador.

Inicialización y acceso a los registros del puerto serie
Habilitación del receptor

El bit de habilitación del receptor (REN) en SCON debe activarse mediante el software para habilitar la recepción de caracteres.

Por lo general esto se hace al inicio del programa cuando se inicializa el puerto serial, los temporizadores, etc.

La instrucción:

SETB REN

activa el REN en forma explícita, o la instrucción:

MOV SCON, #xxx|xxxx B

establece en 1 el REN y restablece en 1 o 0 los otros bits en SCON según se requiera.

El noveno bit de datos

Transmitido en los modos 2 y 3 debe cargarse en TB8 mediante el software. El nuevo bit de datos recibido se transfiere a RB8.