27-01-20

Docente: Martin Isaac Falcon Segovia
Materia: Organización Computacional

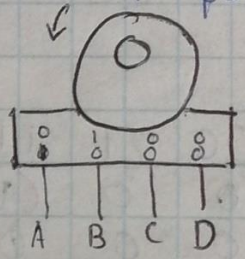
Computadora: Máquina digital que lee información y procesa datos

28-01-20

Máquinas de estado

Lógica combinacional: Máquinas sin memoria que al apagarse, quedan como el principio

Motor a pasos



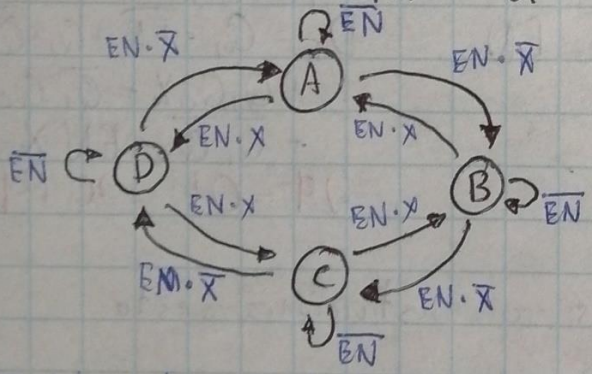
Estados

A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

Lo anterior se hace con la electrónica de estados por depender del estado anterior

Estados: Combinaciones secuenciales

El motor anterior tiene el diagrama de estados siguiente:



$\overline{EN} \cdot \overline{X}$ = Mueve a la derecha
$\overline{EN}$ = Apagar
$EN \cdot X$ = Mueve a la izquierda

Tabla de Estados

Estado presente	Q_1	Q_0	Estado siguiente A (00)	B (01)	C (10)	D (11)	a	b	c	d
A	0	0	A	A	B	D	1	0	0	0
B	0	1	B	B	C	A	0	1	0	0
C	1	0	C	C	D	B	0	0	1	0
D	1	1	D	D	A	C	0	0	0	1

Ecuaciones de salida

Q_1	Q_0	q
0	0	1
0	1	0
1	0	0
1	1	0

$\rightarrow q = \overline{Q_1} \cdot \overline{Q_0}$

Se deduce

$b = \overline{Q_1} \cdot Q_0$

$c = Q_1 \cdot \overline{Q_0}$

$d = Q_1 \cdot Q_0$

Ecuaciones de entrada

Enabled, X $\rightarrow$ Q

Q_0, Q_1

Q_0^*, Q_1^*

$Q_0^* = D_0$

$Q_1^* = D_1$

Mapas de Carnaugh

$Q_1 Q_0$ \ EN·X	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$Q_1 Q_0$ \ EN X	00	01	11	10
00	0	0	1	0
01	0	0	0	1
11	1	1	1	0
10	1	1	0	1

$D_0 = Q_0^*$

$D_0 = Q_0 \cdot \overline{EN} + \overline{Q_0} \cdot EN$

$D_1 = Q_1^*$

$D_1 = Q_1 \cdot \overline{EN} + Q_1 \cdot \overline{Q_0} \cdot \overline{X} + Q_1 Q_0 X + \overline{Q_1} Q_0 EN\overline{X} + \overline{Q_1}\,\overline{Q_0}\, EN X$

29-01-2014

Arquitectura de Von Neumann:
El tamaño de la unidad de datos o instrucciones está conectado por buses con 3 tipos
- Bus de control, de direcciones y de datos e instrucciones
Tiene que hacer varios accesos a memorias

Arquitectura Harvard:
Tiene el CPU conectada a dos memorias diferentes (una con instrucciones y otra con datos)
El tiempo de datos consigue una mayor velocidad de operación

Decodificador de Instrucciones

CISC Complex Instruction Set Computer
RISC Reduced Instruction Set Computer

Secuenciador

Direccionamiento de memorias y periféricos
Su registro de salido es el Program Counter

31-01-20

Program Counter

Este registro apunta directamente a las direcciones distinto al Stack Pointer que funciona como pila

Registro de banderas

Registro especial que es parte de la ALU donde indica cambios realizados por la ALU

- Z - Da cero
- S - Cambio de signo
- O - Overflow
- N - Negativo
- C - Carry
- H - Half carry
- I - Máscara de interrupciones

Ejecucion de las instrucciones

Se divide en fase de búsqueda y en fase de ejecución
Necesita de uno a seis ciclos de reloj

Fase de busqueda

1- Transferir el contenido del Program Counter al registro en el secuenciador

Implementación de FSM

El equipo de Tesla Motors quiere tomar el diseño de luces del antiguo Mercury Capri para su nuevo Tesla Model F. Para no gastar en microcontroladores o sistemas extra han decidido que su mejor opción es diseñar un circuito especial para esta función y han visto a los estudiantes de ICI que son los candidatos ideales para resolver este problema. Para esto han mandado el siguiente diagrama de máquinas de estados para que entiendan qué es lo que solicitan:

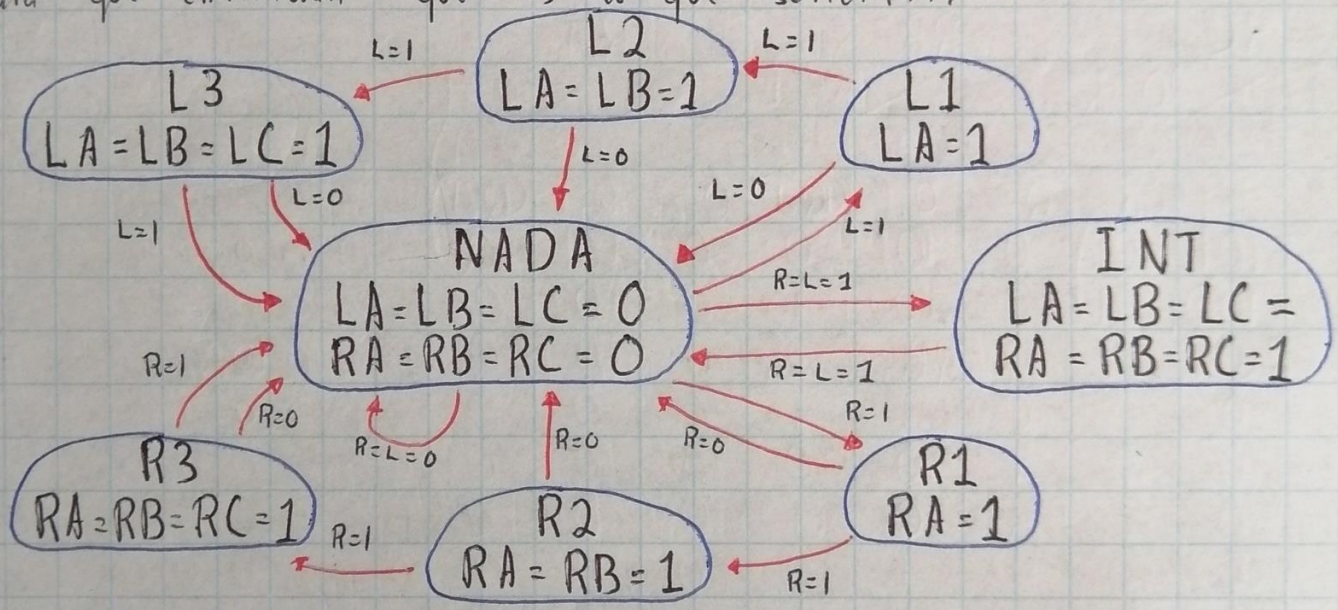


Tabla de estados

E. actual	0 0	0 1	1 0	1 1
NADA	NADA	R1	L1	INT
R1	NADA	R2	NADA	NADA
R2	NADA	R3	NADA	NADA
R3	NADA	NADA	NADA	NADA
L1	NADA	NADA	L2	NADA
L2	NADA	NADA	L3	NADA
L3	NADA	NADA	NADA	NADA
INT	NADA	NADA	NADA	NADA

Asignar valores

N = 0 0 0
R1 = 0 0 1
R2 = 0 1 0
R3 = 0 1 1
L1 = 1 0 0
L2 = 1 0 1
L3 = 1 1 0
I = 1 1 1

Tabla de transición

Entradas		Estado Actual			Estado siguiente			Salidas					
L	R	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	RA	RB	RC	LA	LB	LC
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	1	0	0	0	0	0
0	1	0	0	1	0	1	0	1	1	0	0	0	0
0	1	0	1	0	0	1	1	1	1	1	0	0	0
0	1	0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0

1	0	1	0	0	1	0	1	0	0	0	1	1	0
1	0	1	0	1	1	1	0	0	0	0	1	1	1
1	0	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1	1	1
1	1	0	0	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0

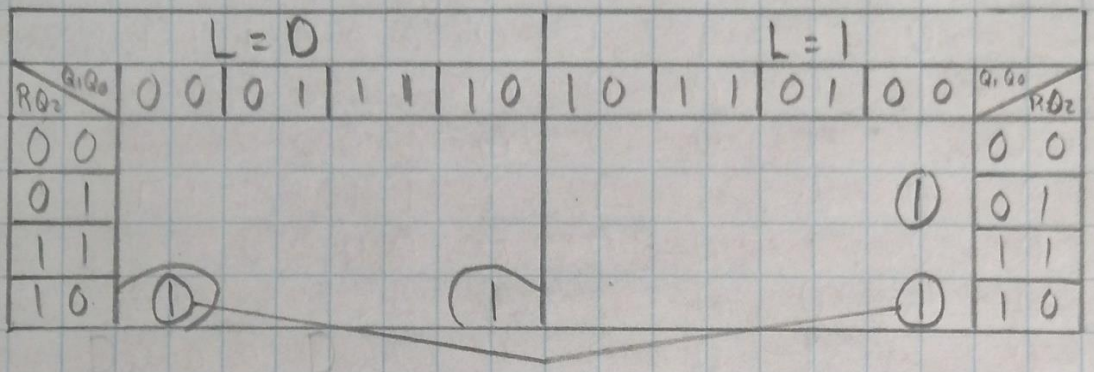
Mapas K (entradas Flip Flops)

L = 0								L = 1								
RQ_2 \ Q_1Q_0	0 0	0 1	1 1	1 0				1 0	1 1	0 1	0 0				Q_1Q_0 \ RQ_2
0 0										(1)					0 0
0 1								(1)		(1)					0 1
1 1															1 1
1 0										(1)					1 0

$$D_2 = L\bar{R}\bar{Q}_1\bar{Q}_0 + L\bar{R}Q_2\bar{Q}_1 + L\bar{Q}_2\bar{Q}_1Q_0$$

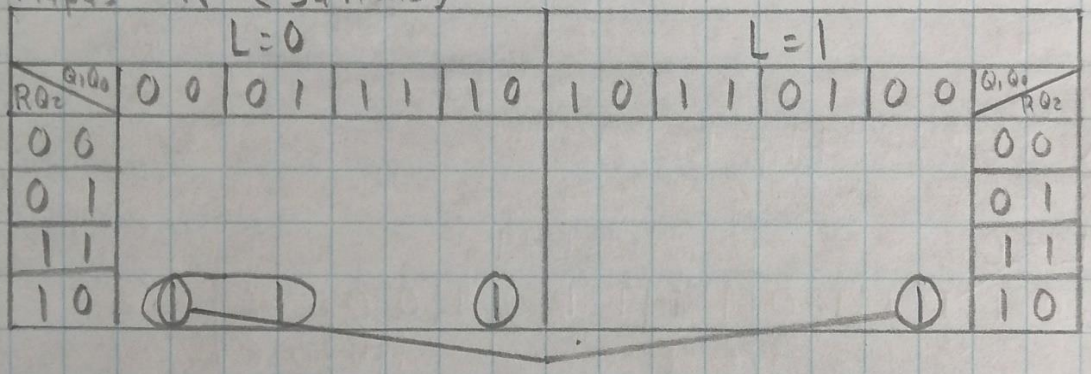
L = 0								L = 1								
RQ_2 \ Q_1Q_0	0 0	0 1	1 1	1 0				1 0	1 1	0 1	0 0				Q_1Q_0 \ RQ_2
0 0															0 0
0 1										(1)					0 1
1 1															1 1
1 0		(1)		(1)						(1)					1 0

$$D_1 = \bar{L}R\bar{Q}_2\bar{Q}_1Q_0 + \bar{L}R\bar{Q}_2Q_1\bar{Q}_0 + L\bar{R}Q_2\bar{Q}_1Q_0 + LR\bar{Q}_2\bar{Q}_1\bar{Q}_0$$

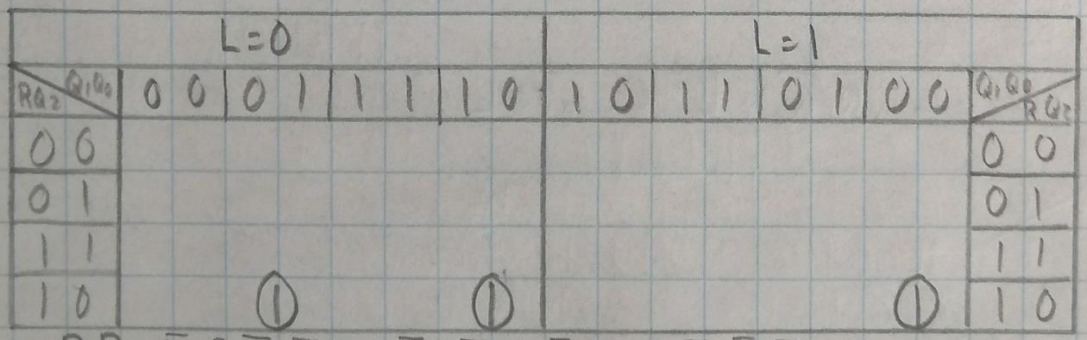


$$D_0 = R\bar{Q}_2\bar{Q}_1\bar{Q}_0 + \bar{L}R\bar{Q}_2\bar{Q}_0 + L\bar{R}Q_2\bar{Q}_1\bar{Q}_0$$

Mapas K (Salidas)



$$RA = \bar{L}R\bar{Q}_2\bar{Q}_1 + R\bar{Q}_2\bar{Q}_1\bar{Q}_0 + \bar{L}R\bar{Q}_2Q_1\bar{Q}_0$$



$$RB = \bar{L}R\bar{Q}_2\bar{Q}_1Q_0 + \bar{L}R\bar{Q}_2Q_1\bar{Q}_0 + LR\bar{Q}_2\bar{Q}_1\bar{Q}_0$$

L=0

RQ_2 \ Q_1Q_0	00	01	11	10
00				
01				
11				
10			①	

L=1

RQ_2 \ Q_1Q_0	10	11	01	00
00				
01				
11			①	
10				

$$RC = \bar{L}R\bar{Q}_2Q_1\bar{Q}_0 + LR\bar{Q}_2\bar{Q}_1\bar{Q}_0$$

L=0

RQ_2 \ Q_1Q_0	00	01	11	10
00				
01				
11				
10				

L=1

RQ_2 \ Q_1Q_0	10	11	01	00
00			1	1
01			1	1
11				
10			1	

$$LA = L\bar{R}\bar{Q}_1\bar{Q}_0 + L\bar{Q}_2\bar{Q}_1\bar{Q}_0 + L\bar{R}Q_2\bar{Q}_1$$

L=0

RQ_2 \ Q_1Q_0	00	01	11	10
00				
01				
11				
10				

L=1

RQ_2 \ Q_1Q_0	10	11	01	00
00				
01			1	1
11				
10			①	

$$LB = L\bar{R}Q_2\bar{Q}_1 + LR\bar{Q}_2\bar{Q}_1\bar{Q}_0$$

L=0

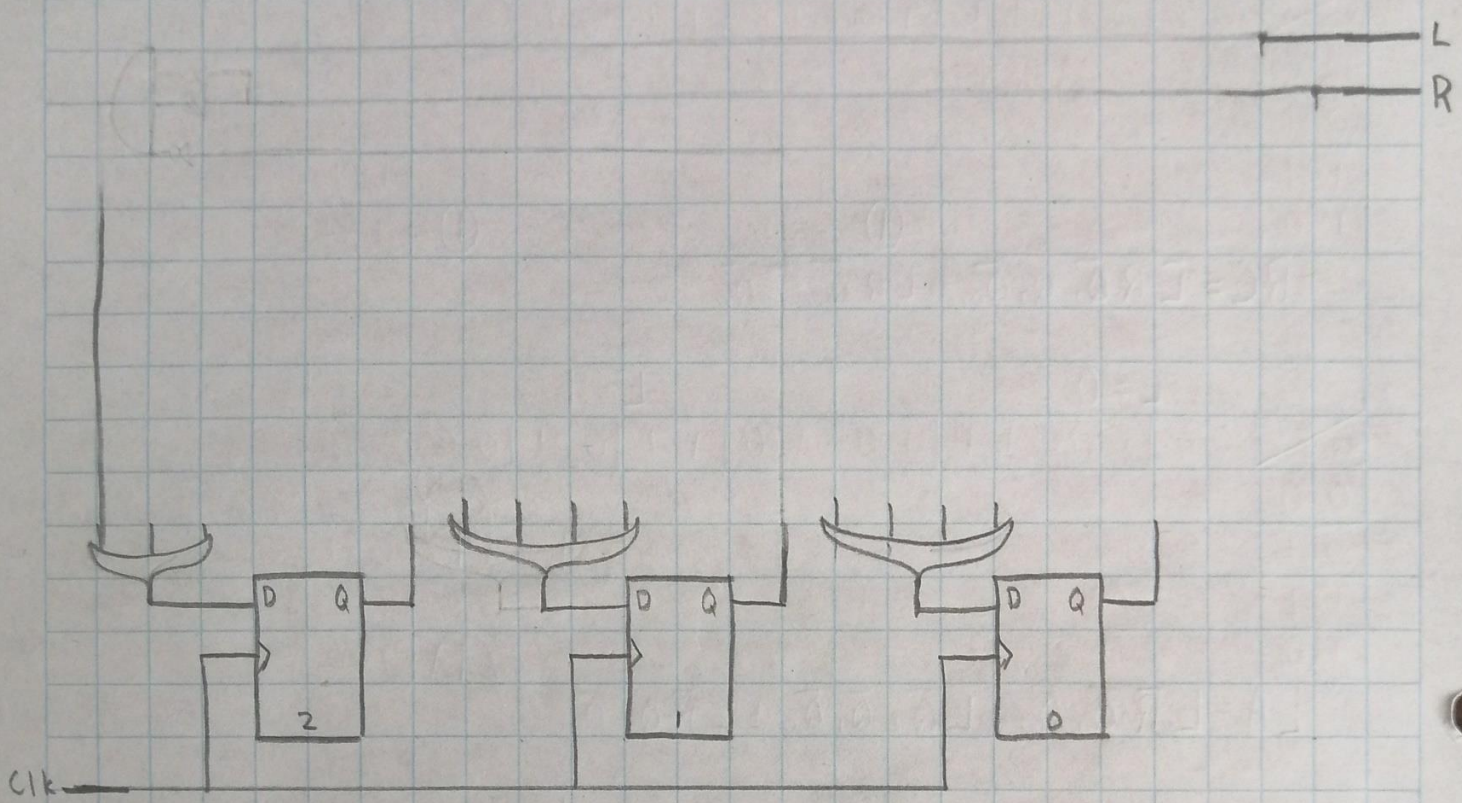
RQ_2 \ Q_1Q_0	00	01	11	10
00				
01				
11				
10				

L=1

RQ_2 \ Q_1Q_0	10	11	01	00
00				
01			①	
11				
10			①	

$$LC = L\bar{R}Q_2\bar{Q}_1Q_0 + LR\bar{Q}_2\bar{Q}_1\bar{Q}_0$$

Circuito



04-02-20

Docente: M en CC Juan Pedro Cisneros S

[redacted]

Evaluación

1er parcial	15%	10% examen
2do parcial	20%	5% tareas
3er parcial	20%	
Prácticas	20%	
Proyecto (se propone)	25%	
	100%	

- Conjunto: Colección de objetos relacionados entre sí. Un conjunto sólo indica los elementos que lo componen, por tanto no es necesario tener un orden y no 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 (no podemos contarlos)
- Los conjuntos infinitos numerables (podemos numerarlos)

- Subconjunto: Colección de elementos que a su vez pertenece a otro conjunto de igual o mayor número de miembros. Como:

Considere:

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

A es un 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.

Ω

- Alfabeto: También llamado vocabulario, es un conjunto finito de símbolos. Debe existir al menos un símbolo en el alfabeto es decir, el alfabeto no puede ser un conjunto vacío. Se le llama comunmente con el símbolo Σ a este conjunto
- Cadena: Secuencia de símbolos hecha con los elementos del un alfabeto. Por eso se dice que una cadena omega ω sobre un alfabeto Σ universal: Σ^*, es decir ω pertenece a Σ^* donde Σ^* es un conjunto formado de todas las cadenas posibles que se puedan hacer con los elementos de sigma. 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

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

Una cadena ω sobre sigma podría ser:

$$10101010$$

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 $L = \{100, 001, 00, 11\}$ se forma con los elementos de $\Sigma = \{0, 1\}$, la cadena $\sigma = 1010$ se forma con los elementos de Σ ($i.e.$, $\sigma \in \Sigma$) 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 obteniendo como resultado otro conjunto de datos de tal proceso

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 hacen una computación, es decir, imaginemos una máquina capaz de seguir una secuencia finita de pasos al introducir un conjunto de datos en ella. Sólo se puede leer un dato en cada paso que se realice, así, los pasos a seguir está dado por los datos a introducir. 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, es decir, que pueda ser descrito mediante una secuencia definida de pasos

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, el hecho es que un autómata y una FSM son lo mismo y podemos utilizar ambos términos simultáneamente. 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. Los autómatas se clasifican en
- Autómata finito determinista
- Autómata finito no determinista

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 del 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 1 y sólo 1 estado inicial

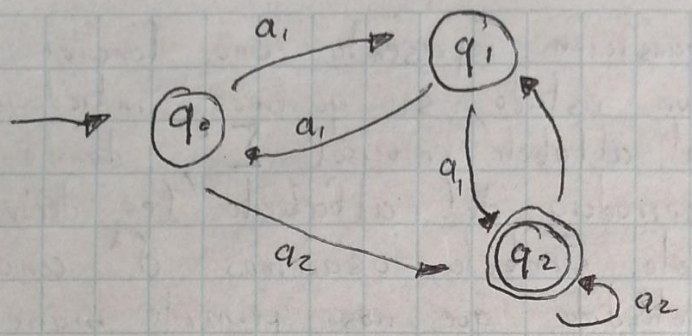
F es un conjunto de estados finales donde $F \subseteq Q$ y q_0 también puede ser final

Σ es un alfabeto finito de entrada

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

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 $\delta(q_i, a) \to q_k$ 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 que q_k, la flecha es etiquetada como a. Si que pertenece a F decimos que la entrada es aceptada.

Debe haber una flecha saliendo de cada estado por cada símbolo $a_0, a_1, a_2, \ldots, a_n$, así, todos los estados tienen el mismo número de flechas saliendo de cada uno de ellos

Así garantizamos que el autómata puede ser llamado determinista (no importa el estado ni símbolo leído, siempre hay una de transición definida)

Para describir 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 queda

	a_1	a_2
→ q_0	q_1	q_2
q_1	q_0	q_2
← 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, y los

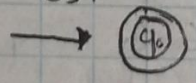
demás no. Una tabla de transición representa una función δ que recibe un símbolo y un estado, si queremos introducir una cadena omega que pertenece al albabeto universal Σ^*, donde el alfabeto universal es la cerradura del alfabeto (es decir, sólo Σ), en lugar de un solo símbolo, usaremos δ^* conocida como función de transición extendido que nos permite manejar una cadena dado que es una función $Q * \Sigma$, y cumple con:

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

Si evaluamos δ^* con un estado $q_i \in Q$ y con $a \in \Sigma$ se comporta de la misma forma que δ.
El primer elemento de Σ^* generalmente es el elemento vacío ε, puede ser el único elemento de nuestro lenguaje $L = \{\varepsilon\}$ o podemos suponer que $\varepsilon \in (\Sigma^* - \Sigma^+)$

07-02-20

El autómata permanece en el mismo estado al introducir epsilon es decir, no cambia de estado. Se comporta como un símbolo neutro para δ^*.
ε puede ser aceptado si y sólo si q_0 también pertenece al conjunto de estados finales. El autómata que sólo acepta el elemento vacío es:



La última propiedad de δ^* nos define cómo evalua 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;

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

Hasta teminar de evalvar la cadena, si el autómata se encuentra en un estado final f se entiende que

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

y la cadena es aceptada.

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

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

Ejercicio

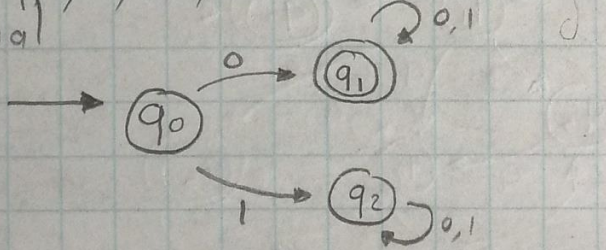
Obtenga un AFD que dé el conjunto de cadenas que inicien en 0

Utilice el alfabeto siguiente

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

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

q_0 = e. inicial



$Q = \{q_0, q_1, q_2\}$

$F = \{q_1\}$

$\delta(q_0, 0) = q_1$
$\delta(q_0, 1) = q_2$
$\delta(q_1, 0) = q_1$
$\delta(q_1, 1) = q_1$
$\delta(q_2, 0) = q_2$
$\delta(q_2, 1) = q_2$

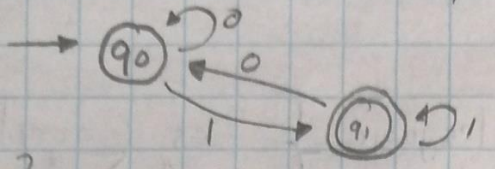
Determine si algona de las siguientes cadenas son aceptadas por el autómata diseñado

a) $\omega_1 = 110\,0101$ No porque $\delta^*(q_0, \omega_1) = q_2$

b) $\omega_2 = 001001$ Sí porque $\delta^*(q_0, \omega_2) = q_1$ ✓

ab ab

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



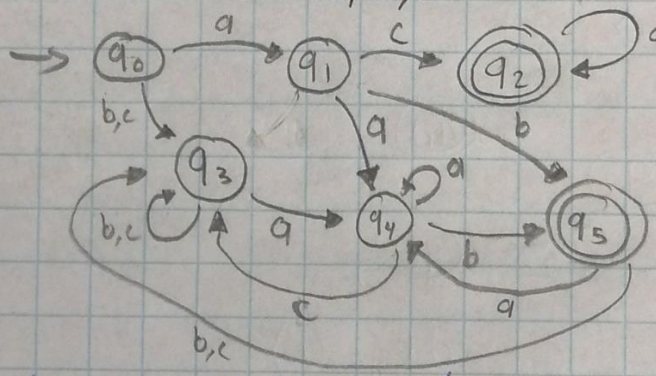
$Q = \{q_0, q_1\}$
$F = \{q_1\}$

$\delta(q_0, 0) = q_0$
$\delta(q_0, 1) = q_1$
$\delta(q_1, 0) = q_0$
$\delta(q_1, 1) = q_1$

10-02-20

Realice un AFD que cumpla con:
- Que obtenga el conjunto de cadenas que inicien en la subcadena ac o que terminen en la subcadena ab

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



Automata finito no determinista

Si con los AFD sólo se puede llegar de una forma de un estado a otro y con un estado inicial, los nd 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 $I \in Q$
F es un conjunto de estados finales donde $F \subseteq Q$

Σ es el alfabeto finito de entrada
δ es la función de transición $Q \times \Sigma \to s$
donde $s \subseteq Q$

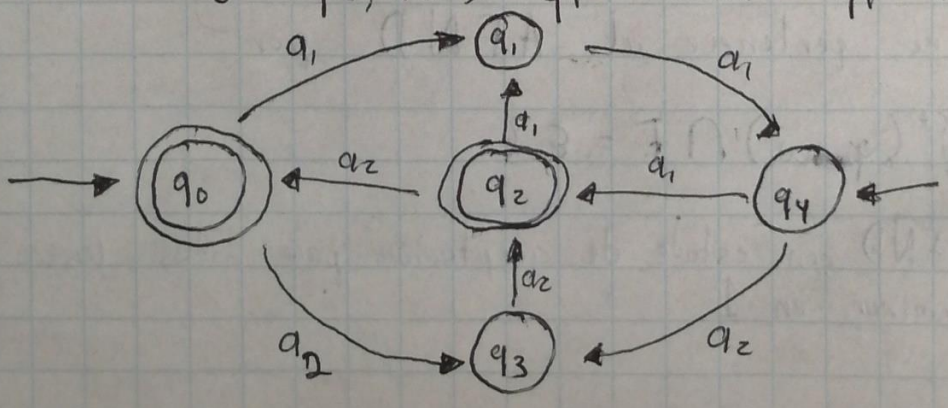
Aparentemente es similar a un AFD pero con diferencias:
- Puede haber más de un estado inicial
- La función δ ahora nos entrega un conjunto tal vez vacío de posibles estados. Precisamente esta es la diferencia de un AFD y un AFND

Cuando todas las transiciones están determinadas en un autómata, es decir, para cada estado, símbolo, existe uno y sólo un estado correspondiente, se tiene un AFD
Si se tiene al menos una transición no definida o indeterminada entonces tenemos un AFND.
En la función de transición extendida δ^* también hay algunos cambios, se puede comenzar en cualquiera de los estados iniciales q_I donde $q_I \in I$ y suma todas las posibles transiciones t que inicien en el mismo estado pero que lleven a estados diferentes a pesar de que a es el mismo entonces podemos definir lo siguiente:

$$\delta^*(q_I, aw) = \sum_{i=0}^{n} \delta^*(t_i, w)$$

Se dice que una cadena es aceptada si

$$\delta^*(q_I, aw) = q_f \quad \text{donde} \quad q_f \in F$$



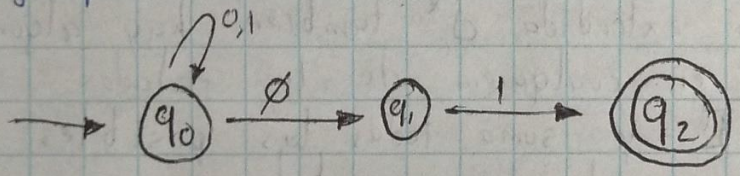
Se puede observar que existe más de un estado inicial y las transiciones no están determinadas:

$$\delta(q_1, a_2)$$
$$\delta(q_3, a_1)$$

Basta una de las condiciones anteriores para considerar al autómata como no determinista. Su tabla de transición es la siguiente

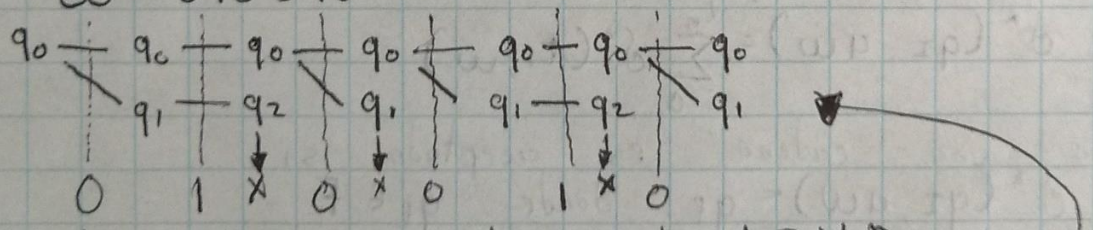
	q_1	a_2
↔ q_0	q_1	q_3
q_1	q_4	∅
← q_2	q_1	q_4
q_3	∅	q_2
→ q_4	q_2	q_3

Ejemplo



Análisis hilos

$\omega = 010010$

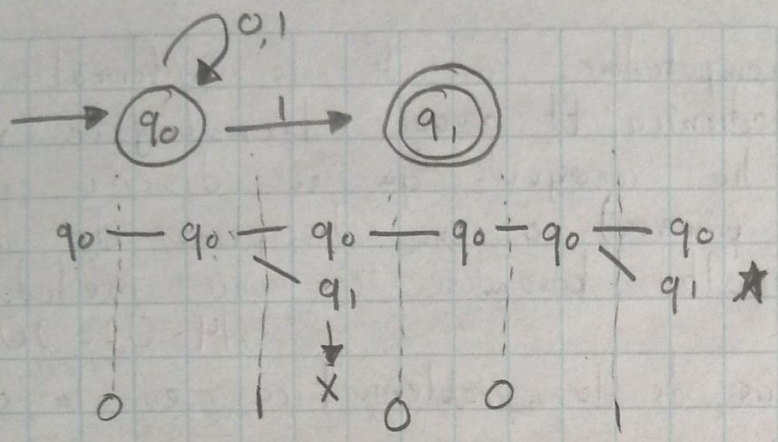


Esta cadena no pertenece al AFND por

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

Obtenga el AFND con estado de aceptación para todas las cadenas que terminen en 1

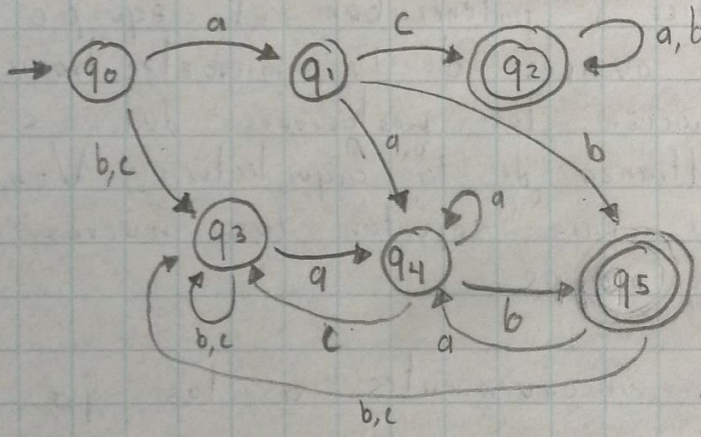
12/02



Con 01001

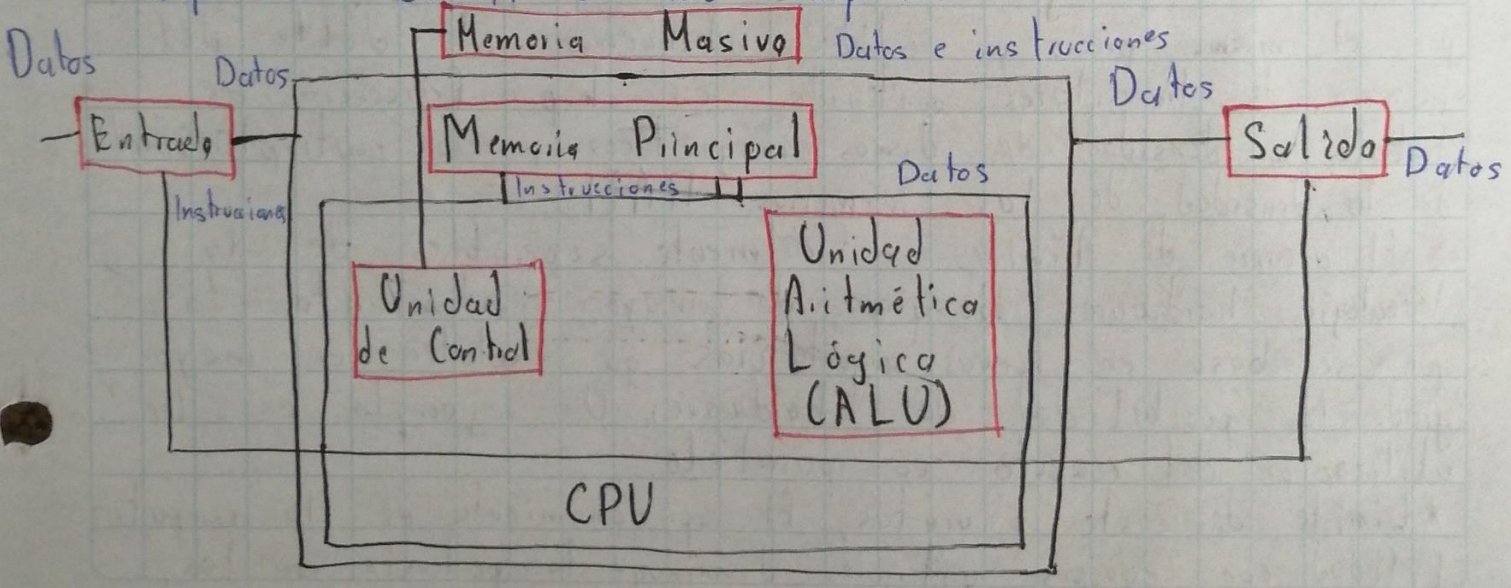
Organización de las computadoras

Implementar un AFND para el ejercicio con las subcadenas ab, ac mencionado previamente



13-02-20

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 ha conseguido que su diseño pase de un arte (en los primeros) a una disciplina de ingeniería que plantea gran dificultad basándose en una metodología

14-02-20

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

1- La independencia entre el hardware y el software y la falta de definición de las funciones de cada uno de ellos. Hasta hace poco, los arquitectos de computadoras procedían de la electrónica y potenciaban el equipo físico aplicando los constantes avances de la microelectrónica sin tener en consideración las prestaciones del sistema lógico

2- El seguimiento a ultranza de la arquitectura Von Neumann no diseñada para soportar los nuevos sist. operativos, lenguajes y apps

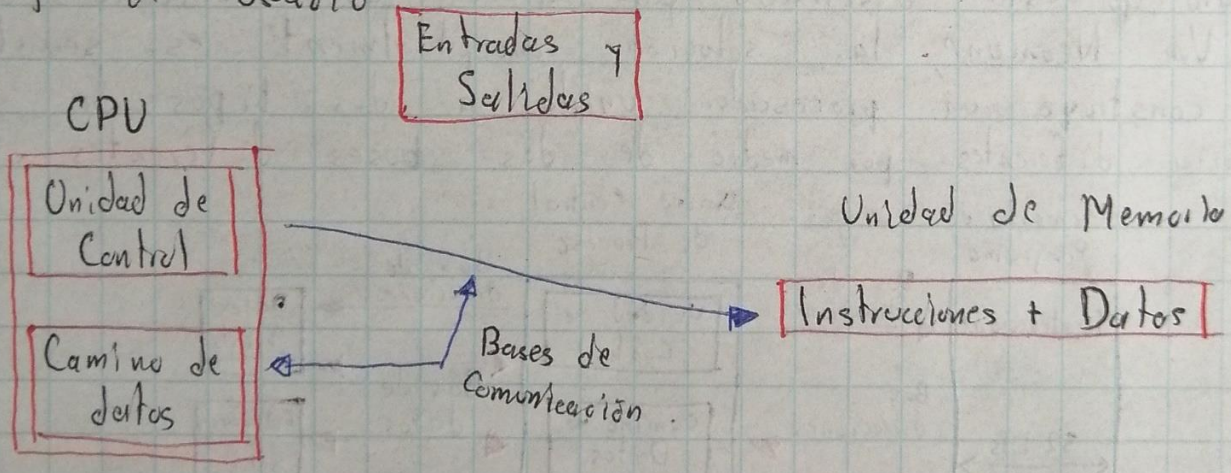
Pero hoy en día los inconvenientes a los que nos enfrentamos son:

* Factor tiempo: En el diseño del equipo físico el tiempo es un factor de gran relevancia. Como ejemplo, cada 3 años por el mismo precio y calidad obtenemos el doble del número de transistores contenidos en chip, frecuencia de trabajo y densidad de discos magnéticos y la cuatripticidad de la densidad de las memorias - RAM
* El alcance de límites difícilmente superables de la tecnología hardware. Búsqueda de mayor rendimiento, la que se basa en nuevas arquitecturas que exploten en mayor grado la posibilidad del hardware. Un ejemplo es la utilización del cómputo en paralelo
* Límite de coste y ventas. El aprovechamiento de la compatibilidad con equipos anteriores disminuye la potencia en los

nuevos equipos físicos
En conclusión el alto nivel de conocimiento requerido del equipo físico y lógico, el factor tiempo, la tecnología hardware insoperable y limitaciones de coste, ventas hacen que la labor del arquitecto de computadoras adquiera un alto grado de complejidad

2.2 Arquitectura Von Neumann (Clásica)

La arquitectura de un computador es la que fue definido por uno de los mejores matemáticos: John Von Neumann. Propuso una arquitectura donde la CPU (Unidad central de procesamiento) está conectada a una memoria que guarda conjuntamente instrucciones (programas) y datos con los que operan los programas. Existe un modulo de entradas y salidas que permiten comunicación con los periféricos externos que maneja el usuario



Si se dispone de un microprocesador que maneja palabras de 8 bits conectado a un bus de 8 bits de ancho que lo conecta con la memoria deberá manejar instrucciones de 1 o más unidades de 8 bits (1 byte)
En esta arquitectura si debemos acceder a una instrucción y/o dato de más de 8 bits tendremos que hacer dos o más accesos a memoria seguidos.
Esta se denomina tipo CISC de su abreviatura "Complex

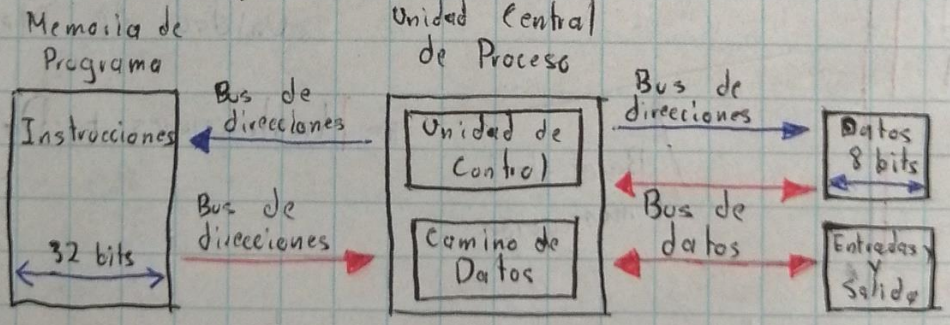
Keil / Mvision
Proteos

Instruction Set Computer. Las instrucciones complejas exigen mucho tiempo de CPU para ejecutarlas y sólo un acceso a memoria que era lento
Por tanto, tiene dos ventajas:
- Longitud de instrucciones está limitada por la longitud de los datos, y 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

17-02-20

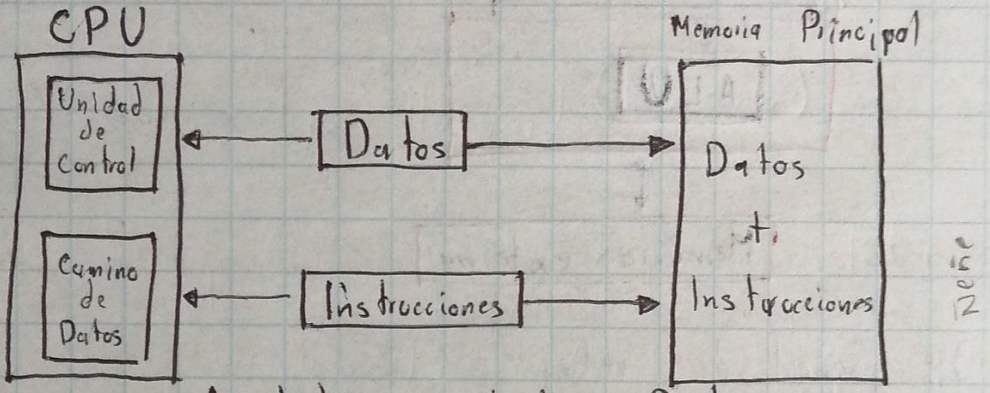
2.3 Arquitectura Harvard (Moderna)

Este tipo de arquitectura modifica el equipo físico, mejoras y nuevas prestaciones en el tiempo lógico.
Un ejemplo en el primer aspecto es la arquitectura Harvard que está diseñada para atacar las debilidades de la Von Neumann, la solución conceptualmente es sencilla.
Se construye 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
Para un procesador de tipo RISC (Reduced Instruction Set Computer), el conjunto de instrucciones y el bus de la memoria del programa pueden diseñarse de modo tal que todas las instrucciones tengan longitud similar que la posición de la memoria y lo mismo con los datos. Además como los

buses de ambas memorias son independientes, la CPU puede acceder a los datos y completar la ejecución de una instrucción y a la vez leer la siguiente instrucción a ejecutar.
Una forma de potenciar el aislamiento entre las instruciones y los datos es la incorporación de memorias caché ultrarápidas que como sucede en los últimos modelos pentium, una guarda los datos que va a precisar la CPU y otra las instrucciones



Arquitecturas actuales Pentium

2.4 Organización de la CPU

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

18-02-20

Diagrama 1: Organización física del computador

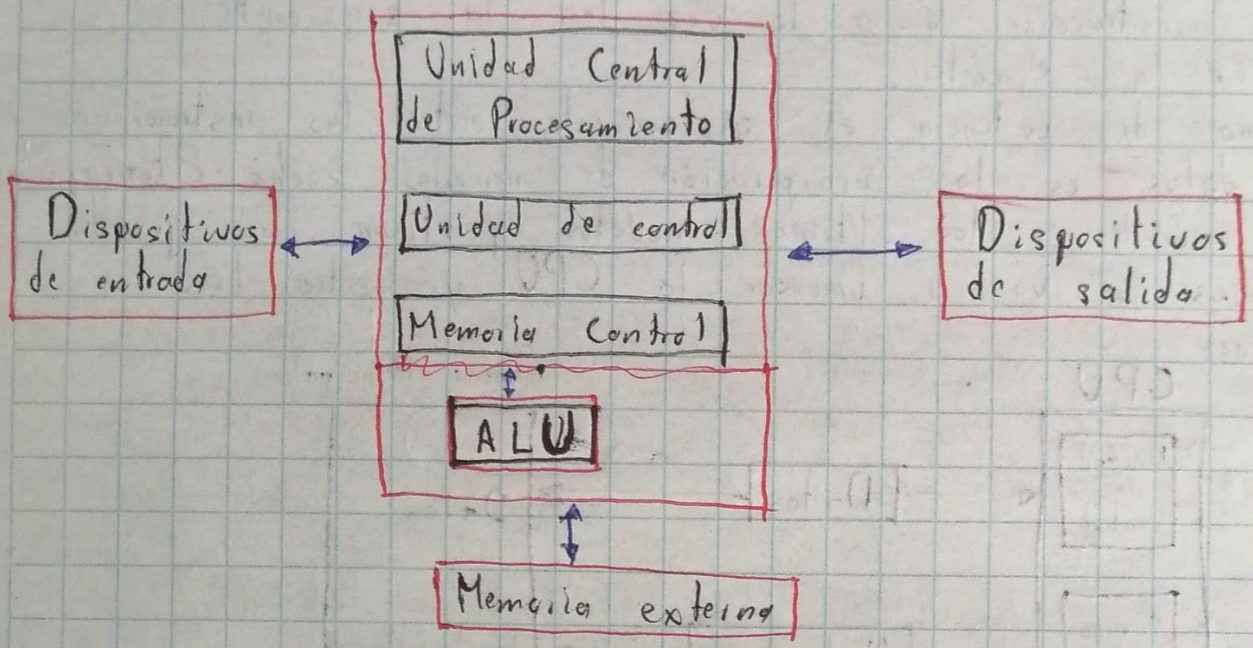


Diagrama 2: Señal de reloj

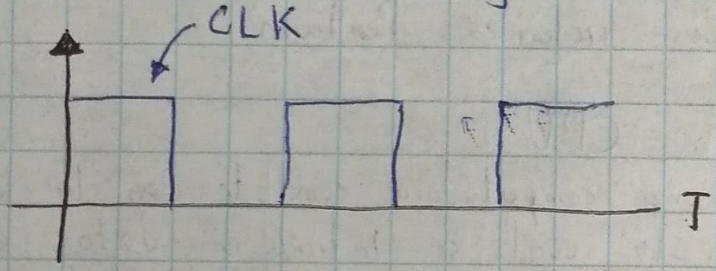


Diagrama 3:

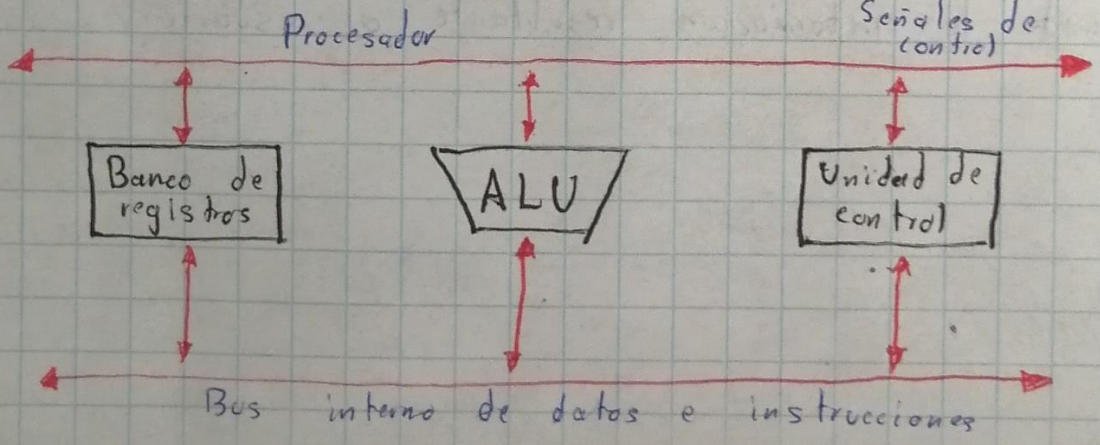


Diagrama 5: Forma de operar en ensamblador.

codop	Rd	Rf

19-02-20

Diagrama 6

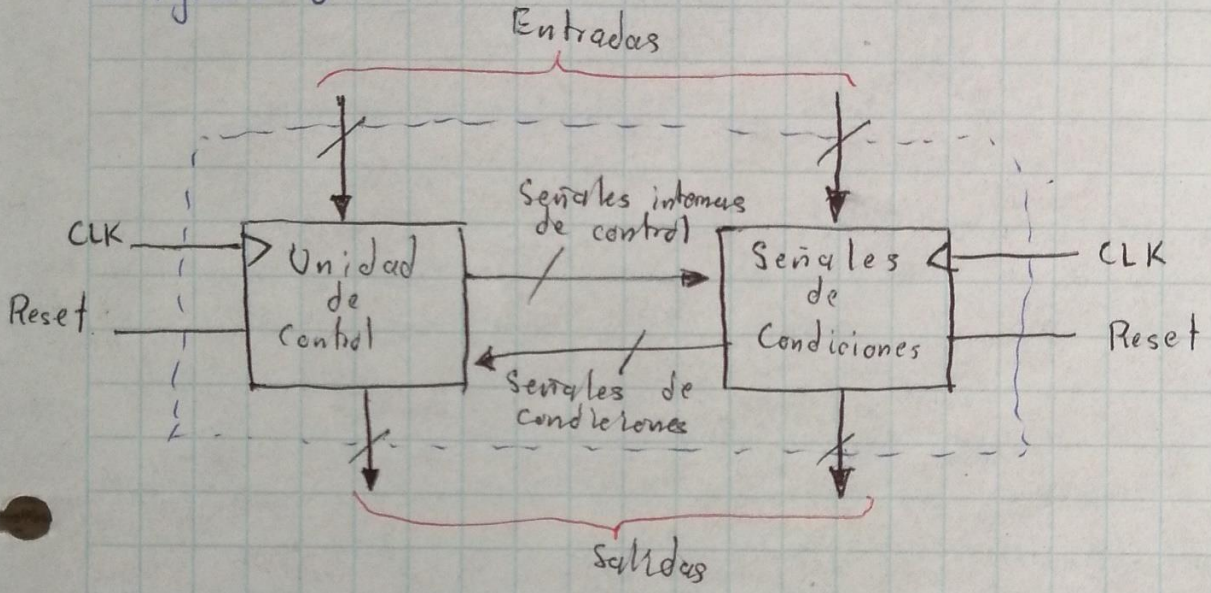
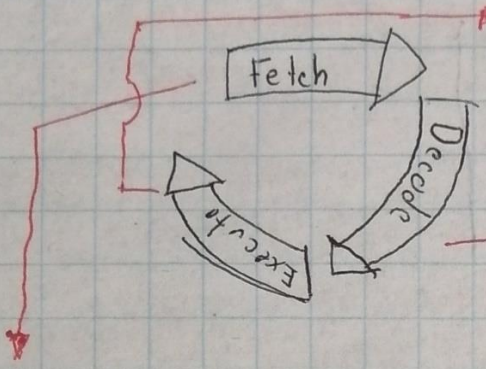


Diagrama 7:



3- Ejecuta la instrucción indicada en el registro de instrucción

2- Decodifica el patrón de bits en el registro de instrucción (IR)

1- Recupera la siguiente instrucción desde memoria y luego incrementa a program counter (PC)

Diagrama 8: Pipeline de 4 etapas

Instrucción	Etapa del pipeline							
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

Diagrama 9:

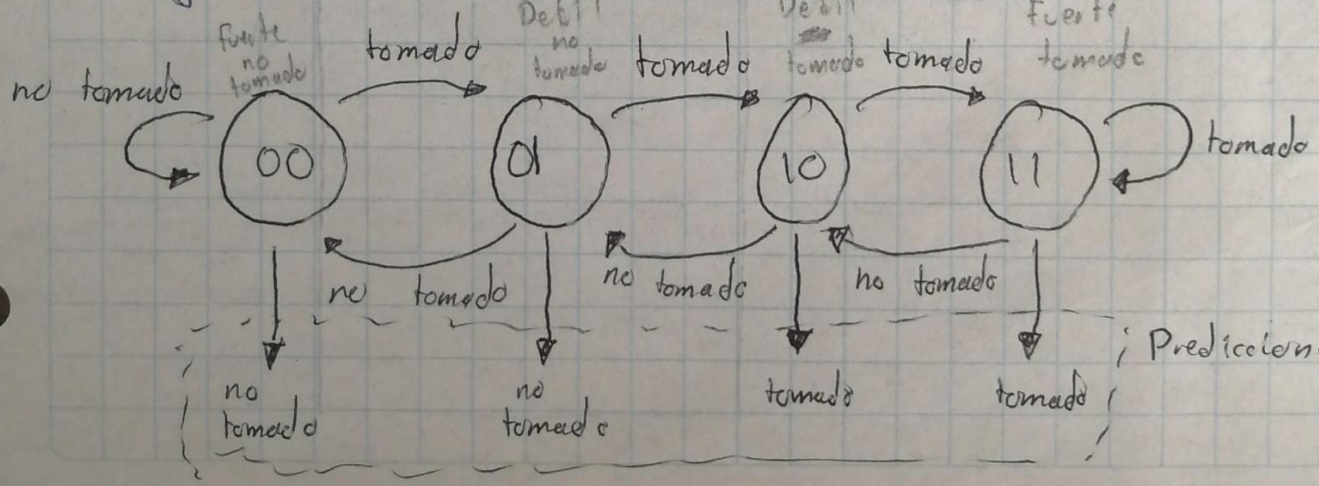


Diagrama 10

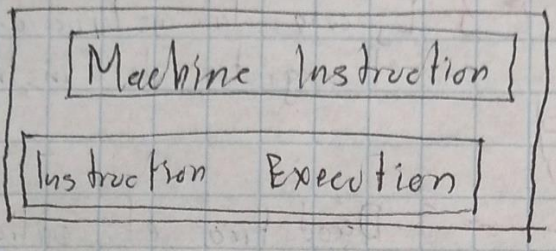


Diagrama 11

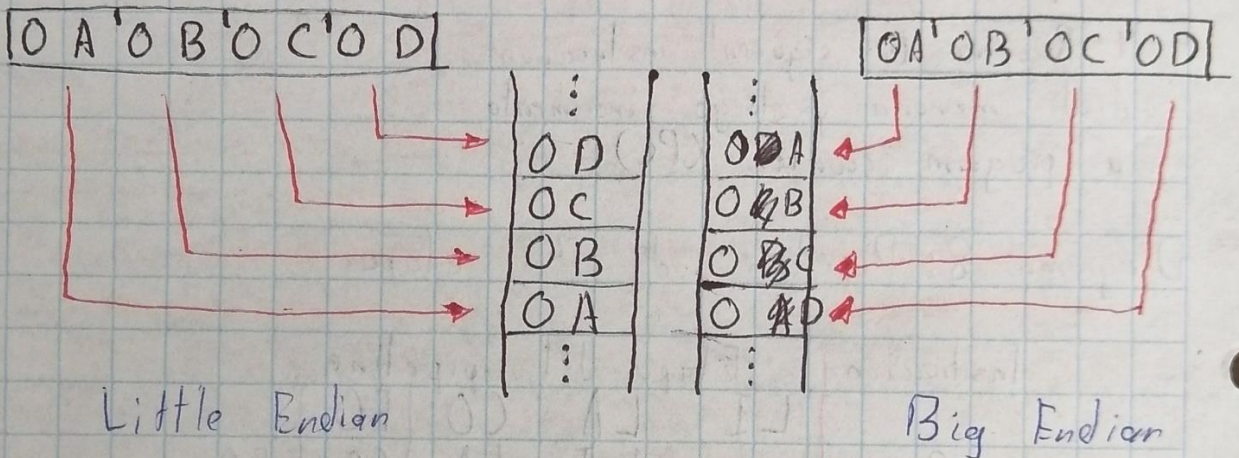


Diagrama 12 Foto en celular