Apunte para el final de orga I

- Von Neumann, Entrada/salida, Memoria –

1 Contenido

2 Von Neumann

2.1 Que es una arquitectura

La arquitectura de una computadora se refiere a los atributos de un sistema visibles para un programador, aquellos que tienen un impacto directo en la ejecución lógica de un programa. Estos atributos incluyen: la ISA, el número de bits que se usan para representar los datos, mecanismos de E/S y modos de direccionamiento de la memoria.

El estudio de la arquitectura se encarga de la estructura y el comportamiento de la computadora.

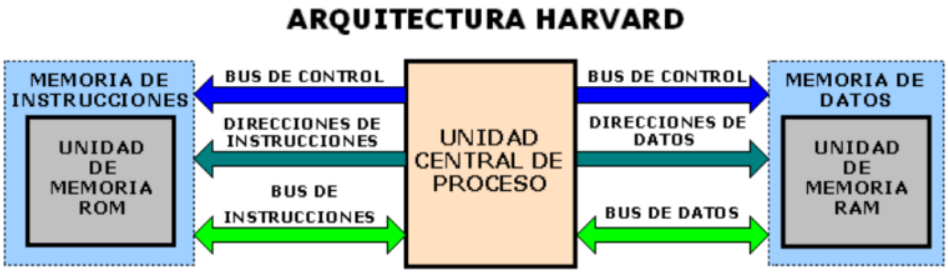
2.2 Características del modelo Von Neumann

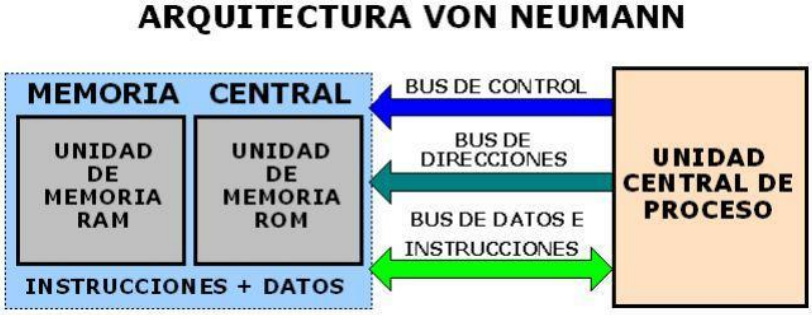
Programa Almacenado: mantiene las instrucciones y los datos en una memoria de acceso aleatorio (RAM). Se diferencia de la arquitectura Harvard en que los datos e instrucciones de programa se almacenan en una única memoria.

Cada celda de memoria tiene una **dirección**: un numero único que la representa.

Los datos y las instrucciones tienen distintos tipos de uso, sin embargo, su estructura no se representa de forma **codificada** (la memoria no puede saber si en una posición tiene un dato o un comando)

Cada programa se ejecuta de **forma secuencial**, a menos que se indique lo contrario mediante instrucciones de control de transferencia.



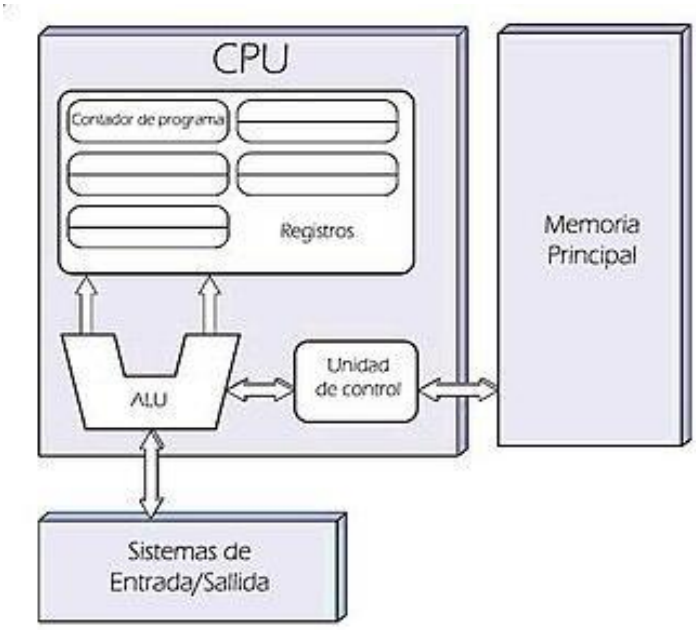


2.3 Estructura del modelo Von Neumann

Consta de tres componentes principales:

- **Unidad central de procesamiento (CPU):** Controla el funcionamiento de la computadora y realiza funciones de procesamiento de datos. Consta de otros cuatro componentes:
  - **Unidad de control:** Se encarga de ejecutar los microinstrucciones que contienen las señales de control de los demás componentes de la CPU.
  - **Unidad aritmético-lógica (ALU):** Realiza las funciones de procesamiento de datos de la computadora.
  - **Registros**: Proporciona almacenamiento interno a la CPU, Pueden ser de propósito específico como el Program Counter o de uso general.
  - **Interconexión de la CPU**: Mecanismo que permite la comunicación entre todos los demás componentes de la CPU
- **Memoria principal:** Almacena los datos y las instrucciones de programas.
- **Sistema de entrada/salida (I/O):** Mueve datos entre la computadora y aparatos externos como sensores o periféricos. Parte de lo que llega de datos a la memoria es resuelto por este subsistema.

La arquitectura tiene la particularidad de que existe un único camino (físico o lógico) desde donde se conecta la CPU con la memoria. En la mayoría de computadoras la velocidad de comunicación ente memoria y procesador es menor a la velocidad a la que puede trabajar el procesador, esto fuerza a la CPU a esperar continuamente a que lleguen los datos desde o hacia la memoria, limitando el rendimiento de la computadora. Esto se conoce como **CUELLO DE BOTELLA DE VON NEUMANN**.



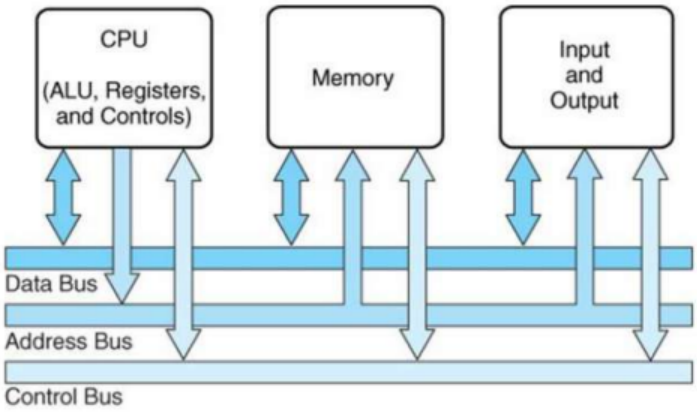
2.4 System bus model

La arquitectura de Von Neumann se puede extender a un modelo llamado “System bus” o **modelo de bus de sistema**.

El bus de datos mueve datos desde la memoria principal a los registros de la CPU (y viceversa).

El bus de direcciones contiene la dirección de los datos que el bus de datos está accediendo actualmente.

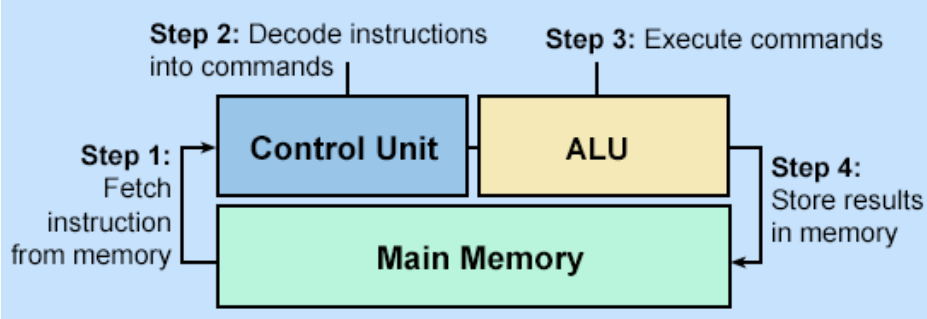
El bus de control lleva las señales de control necesarias que especifican como se llevara a cabo la transferencia de información.



2.5 Ciclo de instrucciones

La arquitectura de Von Neumann ejecuta programas siguiendo el ciclo de instrucción **fetch-decode-execute.** Funciona de la siguiente manera:

1. **Fetch**: Se obtiene la instrucción almacenada en la dirección de memoria que indica el PC y se incrementa el PC, con la ALU o con un incrementador propio; se traen las instrucciones para que la CPU pueda ejecutarlas.
2. **Decode**: Se decodifica la instrucción traída de memoria en un lenguaje que la ALU pueda entender; en caso de que sea necesario, se buscan y obtienen los operandos faltantes desde la memoria y se los coloca en los registros correspondientes para poder realizar la operación.
3. **Execute**: La CPU utiliza los distintos componentes internos para procesar la instrucción correspondiente. De ser necesario almacena el resultado en memoria o en registros.



Ciclo de instrucciones

3 Entrada/Salida

3.1 Interfaz de entrada/salida

Los dispositivos de entrada y salida permiten la comunicación bidireccional entre la computadora y el mundo exterior.

Entrada/salida (I/O) es la transferencia de datos entre la memoria principal y varios periféricos de entrada o salida. Los dispositivos de entrada, como teclado y mouse, nos permiten ingresar datos. Los de salida, como monitor o impresora, permiten obtener información de la computadora.

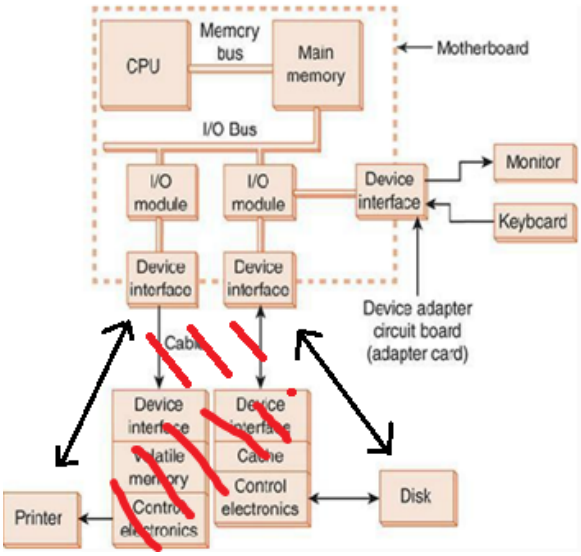
La interfaz de entrada/salida se encarga de esta transmisión de datos, ya que los dispositivos no están conectados directamente a la CPU. La interfaz convierte las señales del sistema de bus a un formato entendible para el dispositivo al cual tiene que llegar. La CPU se comunica con estos dispositivos externos a través de los registros de entrada/salida.

3.2 Arquitectura de entrada/salida

I/O es un subsistema de componentes que transfieren información codificada entre dispositivos externos, la CPU y la memoria.

Los subsistemas de E/S incluyen:

- Direcciones de memoria principal exclusivas para funciones de I/O.
- Buses que proporciona los medios para mover datos dentro y fuera del subsistema.
- Módulos de control dentro de la CPU o dentro de los dispositivos de entrada/salida.
- Interfaces para componentes externos como teclados o discos que permiten la transferencia de datos (por ejemplo, un teclado se conecta a través de interfaces como SATA, USB, etc.)
- Cableado o enlaces de comunicación entre la CPU y los periféricos.



Modelo de configuración I/O

Los **módulos** de I/O se encargan de la transferencia de datos desde la memoria principal hacia una interfaz especifica.

Las **interfaces** están diseñadas para comunicarse con un tipo de dispositivos en particular, como teclados, discos o impresoras. Pueden ser de diferentes tipos como puertos USB, puertos Ethernet, interfaces HDMI, etc. Se diseñan para un propósito específico y para soportar diferentes tipos de periféricos.

3.3 Métodos de acceso a E/S

Los módulos de E/S sirven para comunicar a los dispositivos con el bus de sistema (memoria y CPU).
Los métodos con los cuales los módulos de E/S se comunican con la CPU son cinco:

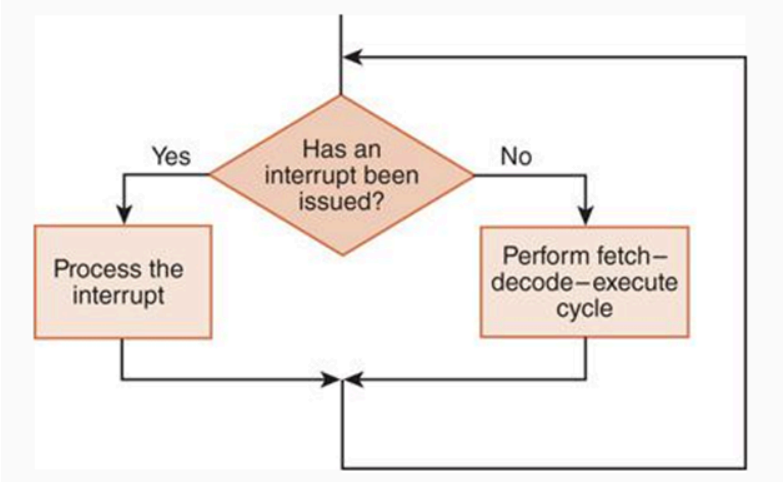
3.3.1 Polling (programmed I/O):

La CPU monitorea continuamente un registro de control asociado con cada puerto de E/S para determinar cuándo hay datos disponibles y cuando puede realizar operaciones de lectura o escritura.
Ventaja: Se puede controlar mediante código el comportamiento de cada dispositivo, el intervalo y las prioridades de polling.
Desventaja: La CPU queda en un bucle continuo llamado *busy waiting* y no hace ningún trabajo hasta que hay algún dato de E/S para procesar.

3.3.2 Interrupciones:

En lugar de estar constantemente monitoreando a los dispositivos de E/S, los dispositivos le indican a la CPU cuando tienen datos para enviar e interrumpen su tarea actual.
La CPU también puede decidir si acepta interrupciones o no dependiendo de la tarea que esté procesando.

Cuando se termina de ejecutar la rutina de interrupciones, se reestablece el estado de la CPU, regresa al programa que había sido interrumpido y continua con el ciclo fetch-decode-execute hasta la siguiente interrupción.



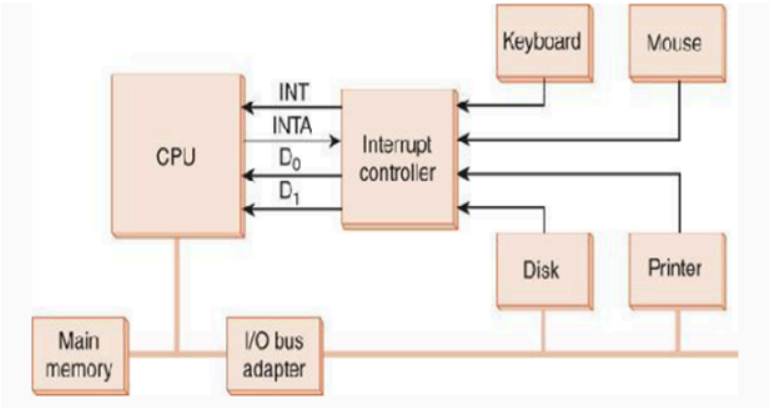
La comunicación entre la CPU y los dispositivos se lleva a cabo mediante un **controlador de interrupciones**, el cual maneja las señales de interrupción de todos los dispositivos de E/S.

Cuando el controlador de interrupciones detecta una señal de interrupción de los dispositivos de E/S conectados, manda una única señal de interrupción que activa una línea de control en el bus de sistema. Por lo general, esta línea de control se conecta directamente a la CPU.

Cada dispositivo de E/S en el sistema tiene acceso a una línea de solicitud de interrupción y el controlador de interrupciones tiene una entrada para cada línea de solicitud de interrupción. Cuando se activa una solicitud de interrupción, el controlador decodifica la interrupción y envía un 1 a la entrada **INT** del procesador.

Cuando el procesador está listo para procesar la interrupción, activa la señal **INTA** (Interrupt Acknowledge) que llega al controlador para que pueda desactivar la señal **INT**.

Cuando dos dispositivos activan su solicitud de interrupción al mismo tiempo, el controlador determina cual tiene prioridad basándose en el diseño del sistema. Las prioridades de interrupción se “cablean” físicamente en el controlador *(hardwired),* lo que las hace prácticamente imposibles de cambiar.



Ventaja: Libera a la CPU de realizar polling y le permite realizar otras tareas, mejorando el rendimiento y optimizando los recursos y tiempos del procesador.
Desventaja: Es menos simple de implementar y gestionar que el método de polling.

3.3.3 Mapeo en memoria:

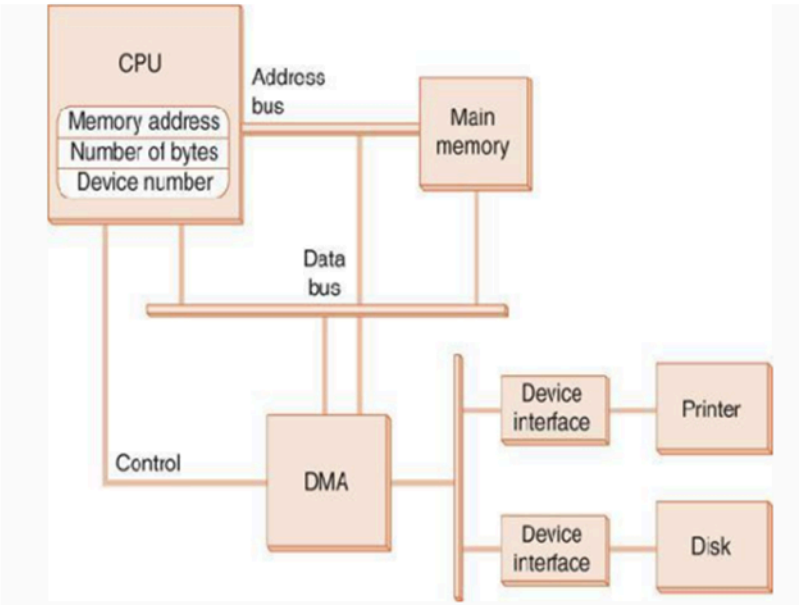
Se reserva una porción de memoria para recibir y mandar datos a los dispositivos. Para usarlo, se usan instrucciones de lectura y escritura en memoria. De esta manera, cada dispositivo de E/S tiene su propio bloque de memoria reservado.
La E/S mapeada a memoria se ve exactamente como un acceso a memoria normal desde el punto de vista de la CPU. Esto significa que se pueden usar las mismas instrucciones para mover datos desde y hacia la E/S y la memoria.

Ventaja: Se simplifica enormemente el diseño del sistema.
Desventaja: Se limita el espacio de direcciones disponibles al ser compartido con los dispositivos de E/S y la memoria principal.

3.3.4 Acceso directo a memoria (DMA):

El proceso de E/S se programa en un chip dedicado llamado controlador de DMA, el cual es mucho más sencillo que una CPU.
La CPU delega la ejecución de instrucciones de E/S en el controlador de DMA.
El controlador de DMA toma el control del bus de sistema (bus de direcciones y de datos) sin intervención de la CPU y transfiere los datos entre los dispositivos de E/S y la memoria. Cuando termina, notifica a la CPU con una interrupción para que pueda procesar los datos transferidos.

Ventajas: Libera a la CPU de la transferencia de datos entre los dispositivos de E/S. Aumenta la eficiencia a la hora de transferir grandes volúmenes de datos y al manejar dispositivos de E/S de alta velocidad como controladores de video.
Desventajas: Tener otro chip que se encargue de la E/S es más costoso y agrega complejidad al diseño de la computadora. Además, el CPU es generalmente más rápido que el chip de DMA, por lo que puede hacer el trabajo más rápido.

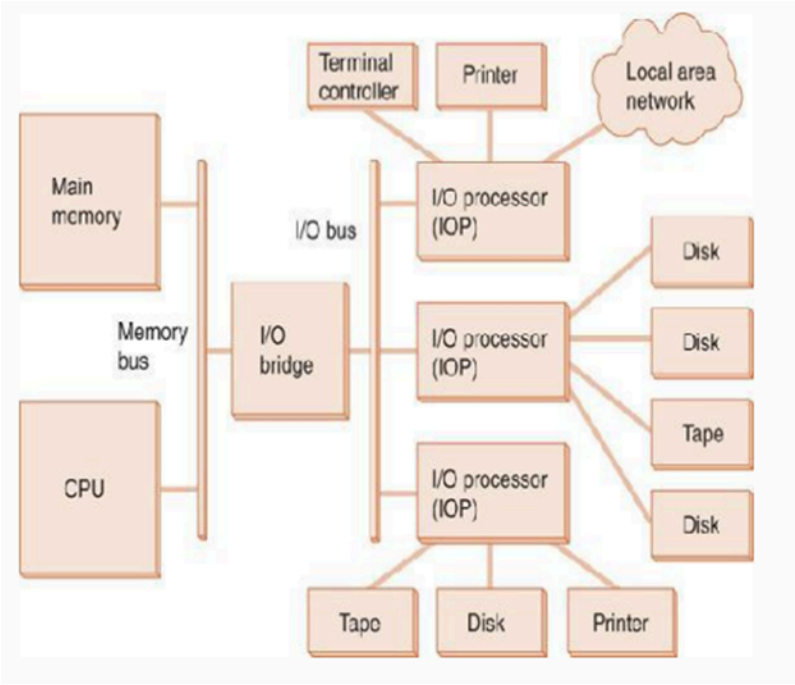


Configuración de DMA

3.3.5 Canal de E/S (Channel-attached I/O):

La mayoría de computadoras que se usan para aplicaciones más críticas (almacenamiento o procesamiento de datos a gran escala o servidores) utilizan un tipo de interfaz de DMA llamada I/O channel.
El canal de E/S funciona como intermediario entre los dispositivos de E/S y la CPU. Los dispositivos se conectan directamente al canal de E/S.

Los canales de E/S son controlados por procesadores optimizados para E/S llamados I/O processors (IOPs). A diferencia de los controladores de DMA, los IOPs tienen la capacidad de ejecutar instrucciones aritmético-lógicas y de saltos condicionales o branches (bifurcaciones) que permiten controlar directamente a los dispositivos de E/S.
Las instrucciones para el control de los dispositivos y la transferencia de datos son puestos en la memoria por la CPU y ejecutados por los IOPs.
Cada IOP puede controlar varias rutas del canal de E/S, permitiendo el control de varios dispositivos de E/S en un solo IOP.



Ventaja: Mejora el rendimiento de sistemas grandes que requieren una mayor eficiencia al gestionar dispositivos de E/S.
Desventaja: Aumenta notablemente el costo y la complejidad de implementación y mantenimiento.

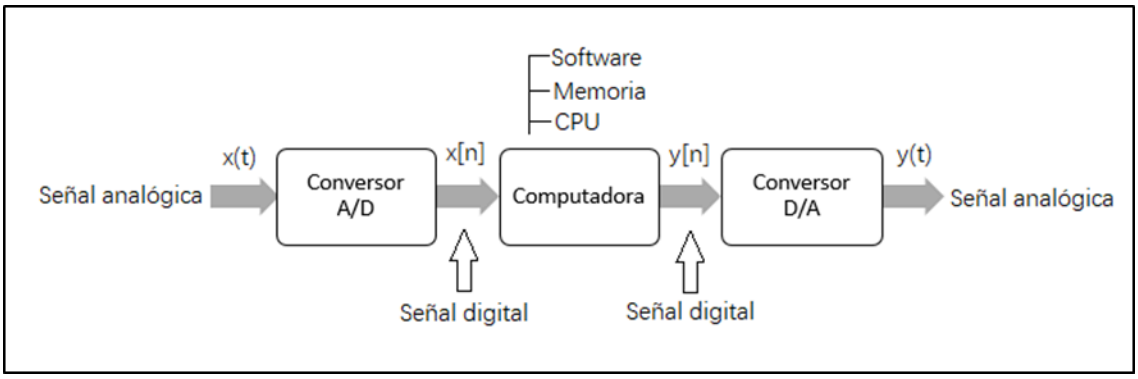
3.4 Comparación de métodos de acceso a E/S

La Velocidad de cada método de I/O depende del hardware dedicado. Lo que no hace el hardware, lo tendrá que hacer el software.

El hardware dedicado a E/S logra una mayor eficiencia por que libera a la CPU de la mayoría de operaciones que se harían con software.

Método de E/S	Hardware	Software	Velocidad
Polling	*	***	*
Interrupciones	**	**	**
DMA	***	*	***

4 Conversión de señales



La conversión de señales se usa para permitir la interacción entre el mundo físico (analógico) y el mundo digital de las computadoras.

Las **señales analógicas** son aquellas que varían de manera continua en el tiempo y pueden tener un rango infinito de valores dentro de un intervalo dado. Son generadas por algún fenómeno, con cierta amplitud y frecuencia, que es representable por una función continua.

Las **señales digitales** son representadas mediante valores discretos (generalmente en forma binaria) donde solamente pueden tener un conjunto finito de valores.

Algunos dispositivos de entrada/salida (sensores, micrófonos, cámaras, etc.) generan **señales analógicas**, estas señales no pueden ser directamente procesadas por una computadora ya que no hay manera de representar los infinitos posibles valores de estas señales. La manera de poder manipular y procesar las señales analógicas en una computadora es convirtiéndolas a formato digital.

Por otro lado, otros dispositivos necesitan que las **señales digitales** almacenadas en forma de datos binarios se conviertan a señales analógicas para ser reproducidas por dispositivos de salida, como parlantes o pantallas.

4.1 Conversión Analógico/digital

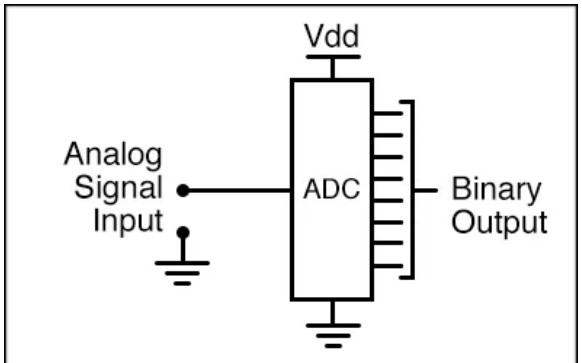


Diagrama de un conversor A/D

La digitalización de una señal analógica consta de dos etapas: Muestreo de la señal y Cuantificación de las muestras.

4.1.1 Muestreo de la señal:

Consiste en tomar muestras del valor de la señal de forma periódica, el periodo entre dos muestras consecutivas se conoce como periodo de muestreo (Ts) o intervalo de muestreo (Δt) y a partir de este valor se puede calcular la frecuencia de muestreo: Fs = 1/ Δt.

Teorema del muestreo: Cuando se muestrea una señal analógica x(t), la frecuencia de muestreo debe ser mayor o igual que dos veces el valor de frecuencia máxima Fmax de la señal x(t), para poder reconstruirla a partir de las muestras.

$$Fs \geq 2*Fmax$$

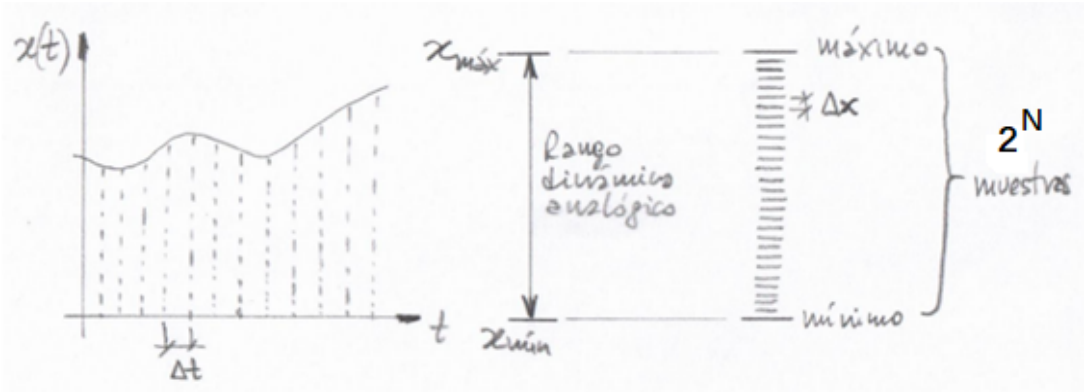
4.1.2 Cuantificación

Consiste en ajustar el valor real de cada una de las muestras al valor más parecido del conjunto de valores posibles para cada muestra.

La cantidad de bits del conversor A/D determina el rango dinámico (Xmin, Xmáx). Cuando se muestrea una señal analógica, se asigna a cada muestra el valor de amplitud más cercano a la amplitud de la onda original. (por ejemplo, un conversor de 16 bits puede cuantificar una muestra y asignarle un valor digital de entre 216 = 65536 valores posibles). Una profundidad de bits más alta proporciona más valores de amplitud posibles, lo que produce un rango dinámico más grande.

El intervalo Δx está dado por la siguiente formula, donde N es la cantidad de bits del conversor A/D:

$$\Delta x = \frac{Xmax - Xmin}{2^N}$$



4.2 Conversión digital/analógico

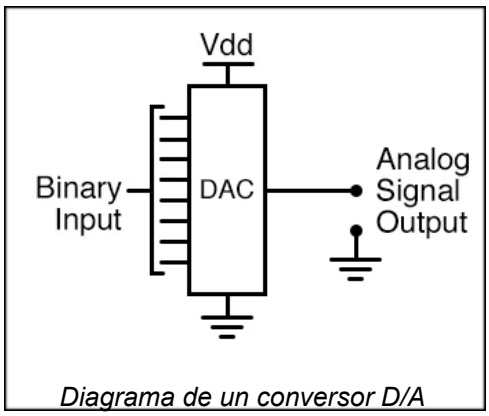


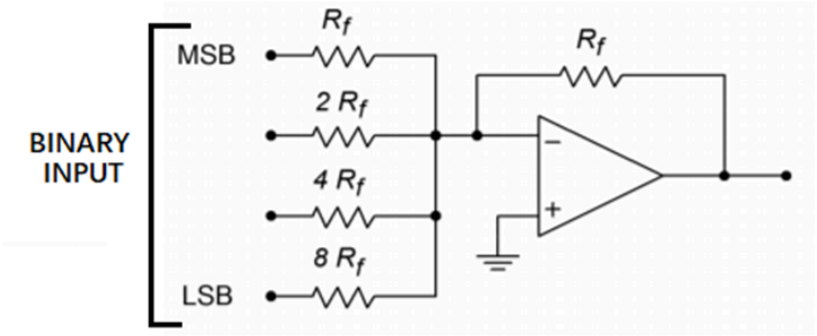
Diagrama de un conversor D/A

El conversor digital/analógico realiza el proceso inverso al que realiza el conversor analógico/digital. Se parte de muestras en formato digital (valores discretos) y estas se deben convertir en una señal analógica (valores continuos).

Para cada combinación de los bits de entrada, el DAC proporciona un valor analógico de salida distinto.

La implementación básica de un DAC es mediante un circuito de resistencias ponderadas usando un amplificador operacional en configuración de lazo cerrado.

Cada bit sucesivo en la palabra digital representa un nivel que es dos veces más grande que el bit anterior. La idea es tomar las señales digitales de entradas y asignarles un peso relativo mediante el uso de resistencias (el bit más significativo (**MSB**) tiene la menor resistencia mientras que el menos significativo (**LSB**) tiene la mayor resistencia, por lo que el peso del MSB es mayor que el del LSB) para luego sumarlas usando el amplificador operacional y obtener una señal analógica entre 0V y un valor máximo dado por la cantidad de bits y el valor de las resistencias.

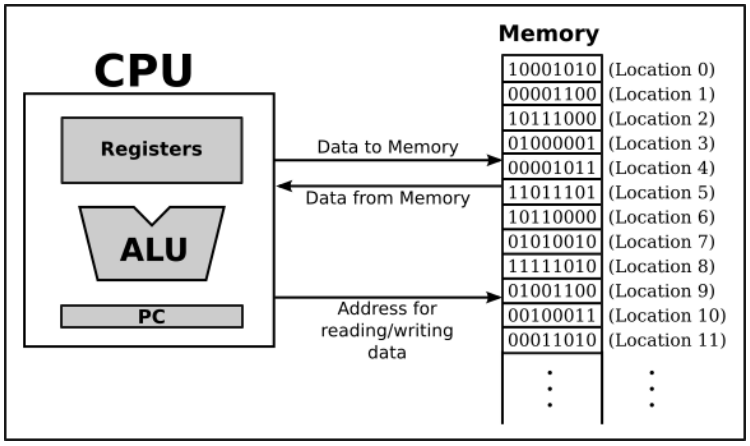


5 Memoria

5.1 Introducción a memorias

Las computadoras basadas en la arquitectura de Von Neumann almacenan los programas y los datos en la memoria.

La estructura lógica de la memoria se puede ver como un **array lineal de direcciones**, desde 0 hasta el tamaño máximo de memoria a la que el procesador puede direccionar.

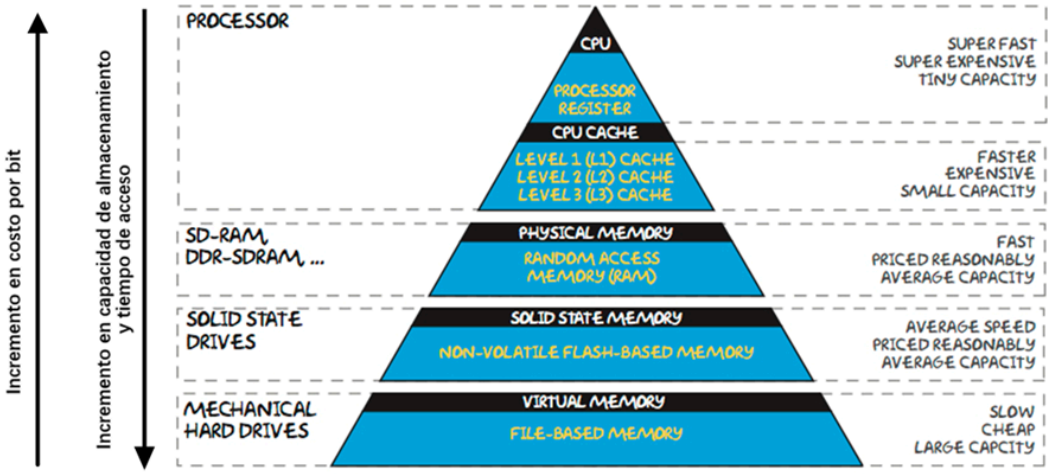


5.2 Jerarquía de memorias

No todas las memorias son iguales, algunas son mucho menos eficientes que otras (y, por lo tanto, más baratas). Para hacer frente a esta disparidad, los sistemas informáticos actuales utilizan una combinación de tipos de memoria para proporcionar el mejor rendimiento al mejor costo. Este enfoque se llama **memoria jerárquica**. Cuanto más rápida es la memoria, más cara es por bit de almacenamiento. Mediante el uso de una jerarquía de memorias, cada una con diferentes velocidades de acceso y capacidades de almacenamiento, un sistema informático puede exhibir un rendimiento superior al que sería posible sin una combinación de los distintos tipos.

La memoria se suele clasificar en función de su "distancia" desde el procesador, con la distancia medida por la cantidad de ciclos de máquina necesarios para el acceso. Cuanto más cerca esté la memoria del procesador, más rápida debería ser. Por lo tanto, se utilizan tecnologías más lentas para las memorias más lejanas y tecnologías más rápidas para memorias más cercanas a la CPU.

Los **registros** son ubicaciones de almacenamiento disponibles en el propio procesador. La memoria **caché** es una memoria de alta velocidad donde se pueden almacenar temporalmente los datos de la memoria principal utilizados con frecuencia. Esta memoria **principal** que suele ser de velocidad media es complementada por otra memoria **secundaria** de mayor tamaño compuesta generalmente por unidades de disco duro o de estado sólido y que contienen datos que la CPU solo puede acceder si se transfieren primero a la memoria principal.



5.3 Tipos de memoria

Existen dos tipos básicos de memoria: ROM (read-only memory) y RAM (random access memory).

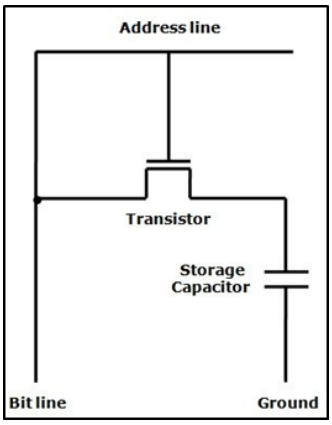
5.3.1 RAM

Es el tipo de memoria que se utiliza para la memoria principal de la computadora. Es usada para almacenar los programas y los datos que necesita la CPU para ejecutar programas.

La memoria RAM es **volátil**, ya que pierde la información almacenada una vez que la computadora deja de recibir energía.

Hay dos tipos generales de chips que se usan para construir memorias RAM: **SRAM** (RAM Estática) y **DRAM** (RAM Dinámica)

5.3.1.1 DRAM



En este diseño, un bit se almacena en un capacitor, de manera que un capacitor descargado representa un 0 y un capacitor cargado representa un 1.

La lectura en DRAM es destructiva, al leer un bit el capacitor se descarga y pierde la información, por lo que necesita regenerar la carga para cada lectura.

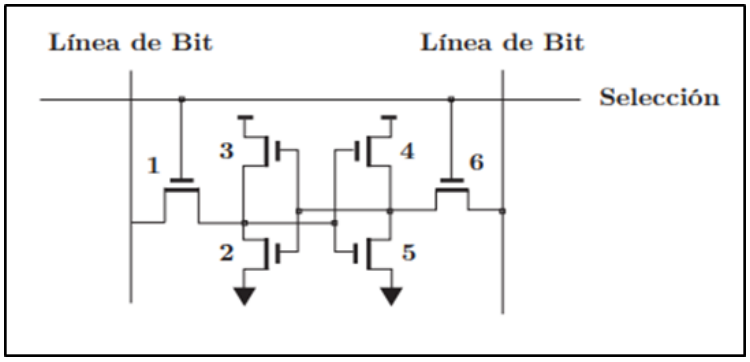
El capacitor junto con el transistor de acceso, consume muy poca energía en comparación con otras tecnologías.

Celda de DRAM

5.3.1.2 SRAM

Las celdas de SRAM están construidas con circuitos de transistores llamados **biestables** (flip-flops o latches).

La lectura es **directa** y **no es destructiva** ya que se puede leer el estado de un bit sin afectar al estado del circuito.
Es más eficiente que DRAM, pero también mucho más costoso en términos de consumo de energía, ya que contiene más componentes por celda.



Celda de SRAM

5.3.1.3 Comparación entre DRAM y SRAM

Ram Dinámica (DRAM)	Ram Estática (SRAM)
Consumo mínimo	Alto consumo relativo
Capacidad de almacenamiento comparativamente alta	Capacidad de almacenamiento comparativamente baja
Costo por bit bajo (almacenar un bit es menos costoso que en SRAM)	Costo por bit alto
Tiempo de acceso lento (regeneración de carga por cada lectura)	Tiempo de acceso más rápido
Si se construye el banco de memoria con DRAM no se aprovecha la velocidad del procesador (cuello de botella)	Si se construye el banco de memoria con SRAM el costo y el consumo de la computadora son altos

5.3.2 ROM

La mayoría de computadoras contienen una memoria de **solo lectura** que sirve para almacenar información crítica (por ejemplo, las microinstrucciones de la unidad de control).
Las memorias ROM **no son volátiles**, por lo que siempre retienen su información, incluso si se apaga el sistema.
Son también usadas en sistemas embebidos o en cualquier sistema en donde no es necesario cambiar su programación.

Existen cinco tipos básicos de memoria ROM: **ROM**, **PROM** (programmable ROM), **EPROM** (erasable PROM), **EEPROM** (electrically erasable PROM) y **memoria flash**.

5.3.2.1 PROM

Es una variación de las memorias ROM en donde la información de un bit depende del estado de un fusible (el valor de fábrica de todos los bits de un PROM es 1; el quemado de cada fusible cambia el valor del correspondiente bit a 0). Una vez programado un PROM, los datos e instrucciones no se pueden cambiar.

5.3.2.2 EPROM

Las memorias EPROM también son programables, pero tienen la característica de que se puede borrar todo su contenido para reprogramarlas (borrar una EPROM requiere un dispositivo especial que emite luz ultravioleta)

5.3.2.3 EEPROM

Las EEPROM eliminan algunas de las desventajas de EPROM ya que no se necesitan herramientas especiales para el borrado de sus datos y solo se puede borrar partes de la información, un byte a la vez.

5.3.2.4 Memoria Flash

Las memorias flash son esencialmente memorias EEPROM con el beneficio adicional de que los datos se pueden escribir o borrar en bloques, eliminando la limitación de un byte a la vez. Esto hace que la memoria flash sea más rápida que EEPROM.

5.4 Memoria caché

Al momento de diseñar la memoria principal, tenemos que lidiar con el problema de la **asimetría** en la tasa de crecimiento de la velocidad de los procesadores con respecto a la tasa de crecimiento de la velocidad de las memorias.

Como actualmente la velocidad de las memorias no son capaces de seguir el ritmo de la velocidad de los procesadores, se utilizan las memorias caché como una **memoria auxiliar** que sirven para que el procesador reduzca el tiempo de acceso a datos ubicados en la memoria principal que se utilizan con más frecuencia.

Como la CPU no tiene forma de saber que datos son más probables que se accedan, se utiliza el **principio de localidad** para transferir los bloques a caché siempre que se haga un acceso a la memoria principal.

5.4.1 Principio de localidad

Para cualquier dato dado, el procesador envía una solicitud a la memoria caché. Si los datos se encuentran en caché (**hit**), pueden cargarse rápidamente a la CPU. En caso contrario (**miss**), la solicitud se envía al siguiente nivel más bajo con respecto a la jerarquía de memoria y comienza nuevamente este proceso de búsqueda.

Si los datos se encuentran en el nivel actual, se transfiere todo el bloque en el que residen los datos a la caché.
La idea es que cuando los niveles inferiores responden a una solicitud de los niveles superiores para acceder al contenido de la ubicación X, envían un bloque con el dato de X y los datos ubicados “alrededor” de X **(…, X-2, X-1, X, X+1, X+2, …)**.

Esto se hace porque es probable que los datos adicionales que incluye el bloque sean utilizados por el procesador en un futuro cercano al pedido del dato original, entonces la transferencia del bloque entero ahorra tiempo de acceso.

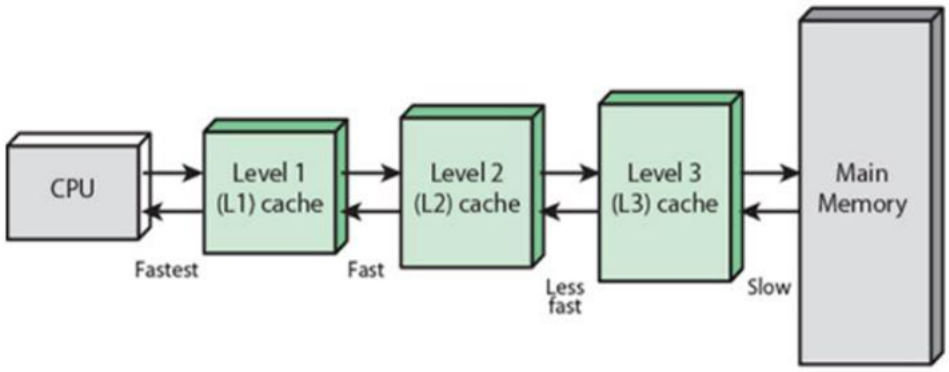
Hay tres formas de localidad:

- **Localidad temporal:** Los ítems accedidos recientemente tienden a ser accedidos de vuelta en el futuro cercano.
- **Localidad espacial:** Los accesos tienden a agruparse en el espacio de la dirección (por ej. Arrays o ciclos)
- **Localidad secuencial:** Las instrucciones o datos tienden a accederse secuencialmente.

5.4.2 Niveles de caché

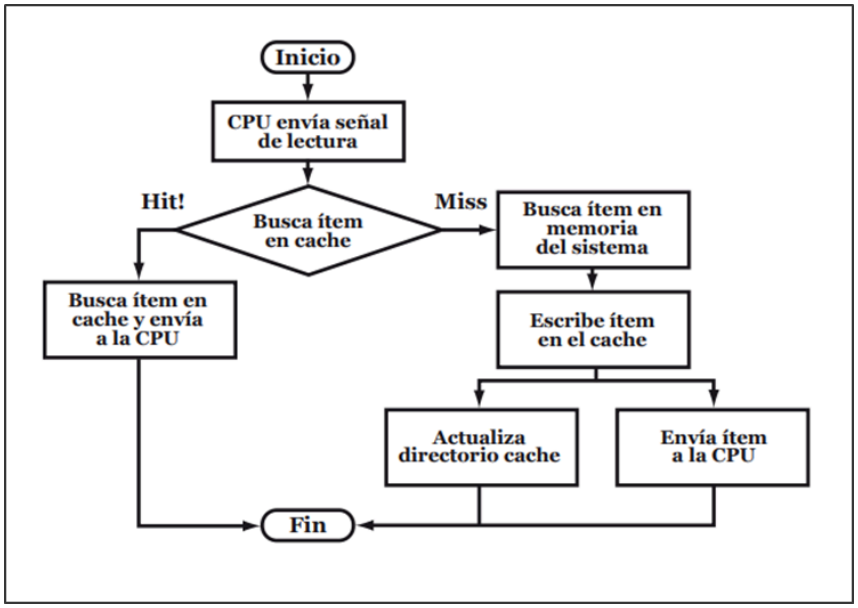
Los datos en la memoria caché se alojan en distintos niveles según la frecuencia de uso que tengan. La información puede transferirse entre los distintos niveles de forma inclusiva (se mantiene una copia de los datos en dos o más niveles) o exclusiva (una vez transferidos los datos, se eliminan del nivel de procedencia)

- **L1 (nivel 1):** Se encuentra dentro del procesador y tiene un tamaño y un tiempo de respuesta menor a los siguientes niveles.
- **L2 (nivel 2):** Puede encontrarse dentro o fuera del procesador dependiendo de la arquitectura. Tiene mayor tamaño, pero es más lenta que caché L1. Puede tener una copia de L1 además de información extra o exclusiva.
- **L3 (nivel 3):** Se encuentra fuera del procesador y es más grande y de acceso más lento que L1 y L2. También puede contener una copia de L2.



5.4.3 Operación de lectura de memoria

Resumen de la operación de lectura en memoria de la CPU:



5.4.4 Hit rate

$$Hit\ rate = \frac{Cantidad\ de\ accesos\ a\ memoria\ con\ presencia\ en\ caché}{Cantidad\ total\ de\ accesos\ a\ memoria} = \frac{hit}{total}$$

El **hit rate** de la memoria caché es la medida de que tan seguido la CPU solicita datos que ya están presentes en la caché en lugar de tener que acceder a la memoria principal.
Idealmente, el hit rate debe ser lo más alto posible ya que indica una utilización eficiente de la caché y una optimización en tiempos de lectura, lo que mejora el rendimiento del sistema.

5.4.5 Algoritmos de reemplazo de contenido

Existen algoritmos para determinar qué datos se eliminan de la memoria caché cuando se necesita liberar espacio para nuevos datos.
La elección del algoritmo depende de factores como el tamaño de la caché o la complejidad de implementación y tiene un impacto directo en el **hit rate**.
Algunos algoritmos son:

- **LRU** (Least recently used): Elimina los datos que no han sido accedidos durante más tiempo. Se corresponde con el principio de localidad temporal.
- **LFU** (Least frequently used): Elimina los datos que han sido accedidos con menos frecuencia.
- **Random:** Elimina los datos aleatoriamente.
- **FIFO;** Elimina los datos más antiguos de la caché.

6 BIBLIOGRAFIA

Null L., Lobur J. – The essentials of computer organization and Architecture. Capitulos 6 y 7.

All About Circuits - Introduction to Digital-Analog Conversion - Link.

David J. Eck - Introduction to Programming Using Java – Sección 1.1 (The Fetch and Execute Cycle)

James M. Fiore – Operational Amplifiers & Linear Integrated Circuits:

Theory and Application – Sección 12.4 (DACs).

Andrew Scholer – Welcome to CS – Sección 6.5 (Memory Hierarchy)

The FurfiOS Corporation - Resumen Final de Orga 1 (Parte 2).

Ramiro Sánchez Posadas y Martino Simón – Q&A para final de Orga I.

Diapositivas de las clases teóricas del Dr. Marcelo Risk. ☺