Arquitectura de Von Neumann

Conceptos fundamentales:

Computadora: Es una máquina digital electrónica programable para el tratamiento de la información, capaz de recibirla, operar sobre ella, y suministrar los resultados de tales operaciones.

¿Cómo se diseña una computadora y qué puede hacer?

Arquitectura de computadora:

Se refiere a los aspectos lógicos de la implementación del sistema tal como los ve el programador. Se enfoca en la estructura y el comportamiento de la computadora, como el conjunto de instrucciones, los modos de direccionamiento y el diseño lógico general.

La arquitectura se da en combinación del hardware y la ISA, instrucciones fundamentales que existen para poder comunicarse con el hardware, con ellas es posible correr el software sobre la máquina.

- Conceptual
- Visible para el programador (lenguaje máquina, instrucciones)
- Especificación de nivel alto

- Atributos que tienen un impacto directo en la ejecución lógica

Ejemplos:

Tipos de datos, tipos de registros, tipos de direccionamiento.
Set de instrucciones, bits utilizados para representar datos…
Conjunto de instrucciones RISC

¿Cómo funciona la máquina?

Organización del computador:

Se refiere a cómo se implementa físicamente los atributos de la arquitectura, es decir, los detalles del hardware que permiten que las especificaciones de la arquitectura se cumplan.

- Práctico
- Oculto al programador (enfoque al hardware interno)
- Nivel detallado (implementación concreta)

Ejemplos :

Señales de control, tecnología de la memoria, ALU, buses de control..

- Ambos son no son temas independientes, se complementan entre sí para el correcto funcionamiento del computador,
- No hay una clara distinción.

- “Una familia de modelos de computadora, todos con la misma arquitectura pero con diferencias en la organización”, como consecuencia muchas veces se permite la retrocompatibilidad en programas antiguos como es el caso de la familia x86 de Intel.

Modelo de Von Neumann

La Arquitectura de Von Neumann, también conocida como modelo de Von Neumann, es aquella arquitectura de procesador basada en la descrita por el matemático y físico hunagro, John von Neumann.
Antes la idea de programar estaba relacionada con el trabajo de conectar cables, lo que lo volvía más en una tarea de ingeniería electrónica. Cada vez que había que calcular algo distinto había que reconectar todo. La idea de almacenar programas fue publicada y publicada por John, quien participó en las últimas etapas del proyecto ENIAC. Aunque no está claro que se le haya ocurrido a él primero.
Debido a que **John W. Mauchly y J. Presper Eckert** documentaron la idea como base de la EDVAC, sin embargo no lo publicaron debido al contexto histórico, ya que el trabajo que mantenían era secreto.

Características:

Actualmente todas las computadoras con programas almacenados utilizan la arquitectura Von Neumann, todas tienen las siguientes características:

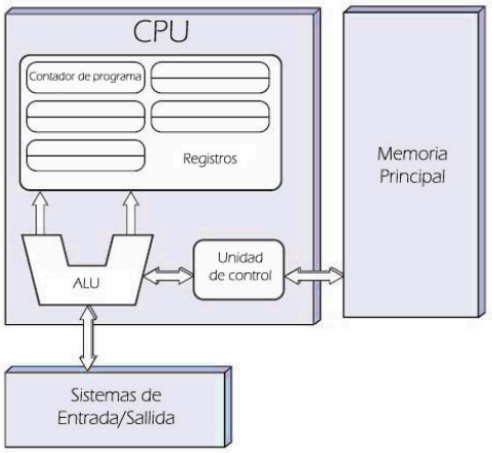
1. Datos y programas se almacenan en una misma memoria sobre la que se puede leer y escribir. Que se van a usar para operar instrucciones que uno va a ejecutar
2. La operación de la máquina depende del estado de la memoria
3. Procesamiento de instrucciones de forma secuencial: siguiendo el orden en que están almacenadas en la memoria principal, salvo que una instrucción específica (como un salto o una bifurcación) indique lo contrario.
4. El contenido de la memoria es accedido a partir de su posición (sin importar su tipo)
5. Cuello de botella Von Neumann: existe un único camino, no es posible transferir datos e instrucciones entre CPU y a la memoria principal al mismo tiempo.
6. Componentes del computador:

    - CPU:
        - ALU: Se encarga de operaciones aritméticas y lógicas bit a bit.
        - CU: Se encarga de dirigir las operaciones del procesador y cómo responder a las instrucciones del programa.
            - IR (Instruction Register): Almacena la instrucción que se está ejecutando actualmente o está siendo decodificada
        - Registros: Son unidades de almacenamiento pequeñas
            - PC (Point Counter): Indica en qué parte del programa está siendo procesada la secuencia.

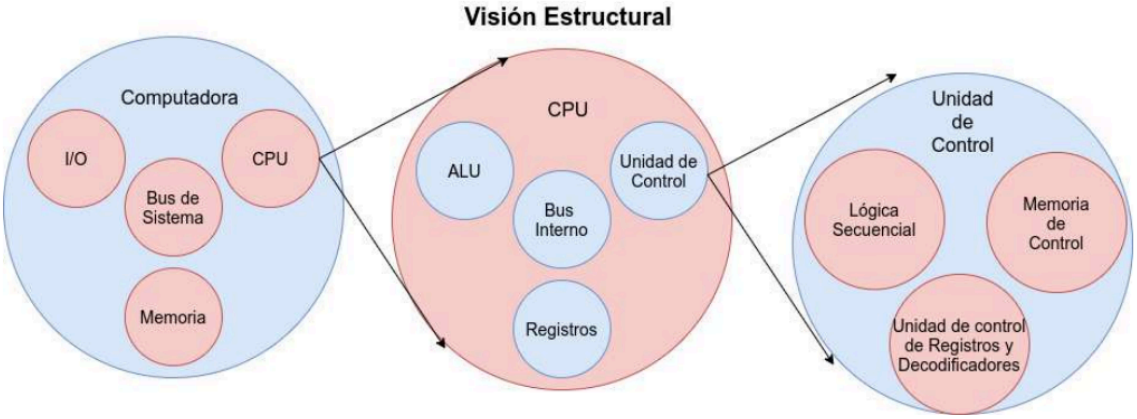
Aumenta luego de hacer fetch de una instrucción.

- Memoria principal:
  Almacena tanto datos como instrucciones.

- Mecanismos de entrada y salida:
  De esta forma podemos ingresar y extraer datos desde su entorno.



- Interconexión del sistema: La conexión suele ser por medio de un bus del sistema, que consiste en una serie de cables a los que se unen todos los demás componente



Ciclo de ejecución de Von Neumann:

La CPU opera instrucciones indicadas en la memoria a través del ciclo de instrucción.
Esta arquitectura ejecuta programas en lo que se conoce como el ciclo de ejecución de Von Neumann, el cual también se lo llama el **ciclo de fetch-decode-execute**. Este se divide en tres partes:

➢ **Fetch:**
- Se obtiene la instrucción almacenada desde la dirección de memoria que indica el PC
- Se incrementa el PC, con la ALU o con un incrementador propio
- Se traen las instrucciones para que la CPU pueda ejecutarlas.

- **Decode:**
  - Se decodifica la instrucción traída en un lenguaje que la ALU pueda entender
  - En caso de que sea necesario, se buscan y obtienen la información faltante desde la memoria y se los coloca en los registros correspondientes, para poder realizar la ejecución.
- **Execute:**
  - Se realiza cuando ya se tiene la instrucción y todo lo necesario para poder ejecutarla
  - La ALU realiza la operación que corresponda, y si es necesario se coloca el resultado en donde sea indicado por la instrucción.

Relación con la Maquina de Turing:

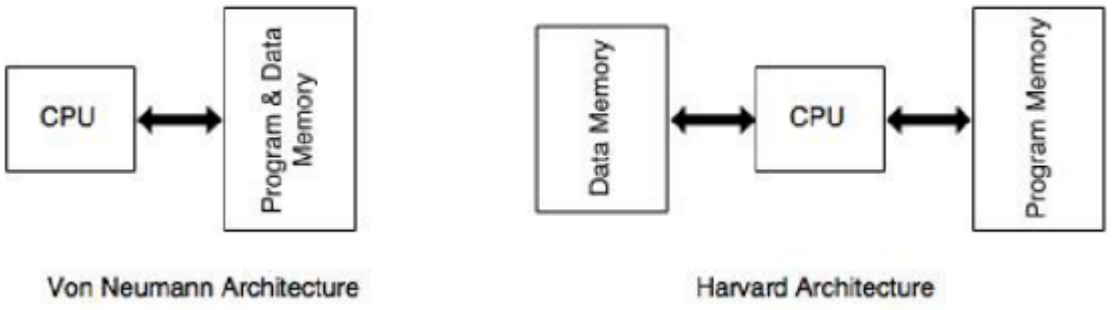
La Máquina de Turing es una abstracción matemática que se le acredita a Alan Turing. Describe un modelo de computadora hipotético, capaz de ejecutar cualquier algoritmo computacional.Esta consiste en una cinta infinita dividida en casillas y reglas de transición que gobiernan el comportamiento de la máquina.

El concepto clave, es su capacidad para ejecutar algoritmos mediante un proceso iterativo de leer, escribir y cambiar de estado, siguiendo reglas definidas. Turing demostró que este modelo era capaz de realizar cualquier cálculo computacional que se pueda realizar de manera algorítmica, **estableciendo así los límites de lo que es computable**.

La arquitectura propuesta por Von Neumann se inspiró en gran medida en los principios establecidos por Turing. Incorpora la idea de almacenar tanto datos como programas en la memoria de la computadora, lo que permite a la CPU acceder a ellos y ejecutar instrucciones según sea necesario, lo que **refleja el concepto de la Máquina de Turing de tener un conjunto finito de reglas y datos que gobiernan el proceso de cálculo de una manera secuencial.**

Comparación con el modelo Hardware:

Como diferencia, tenemos que las instrucciones y datos se almacenan en diferentes memorias, como consecuencia tienen conjuntos de direccionamientos distintos que no interfieren entre sí. Volviéndolo así más seguro y rápido, sin el problema de cuello de botella, sin embargo más costoso.



Von Neumann Architecture

Harvard Architecture

Historia del computador

Generación 0: Sistemas mecánicos y padre de la computación (hasta 1945)

Por los años 1500, las personas utilizaban ábacos para calcular y guardar el resultado en números romanos. Con la aparición del sistema decimal, aparecieron las primeras calculadoras mecánicas, como la Pascalina (Blaise Pascal). Aunque su diseño era limitado, sólo podían resolver operaciones simples como sumas y restas, y además era de fabricación costosa significó un gran avance en la mecanización del cálculo. El diseño mecánico no dejaba margen de error humano, como pasaba con el cálculo manual.

El siguiente gran avance llegó con **Charles Babbage**, considerado el "padre de las computadoras". Su diseño de la **Máquina Diferencial** fue una calculadora mecánica creada para resolver **funciones polinómicas** con gran precisión. Más tarde, desarrolló la **Máquina Analítica**, un concepto revolucionario que no solo podía realizar **cálculos algebraicos**, sino que también introdujo elementos fundamentales para las computadoras modernas. Este diseño incluía componentes como **memoria** para almacenar datos, mecanismos de **entrada y salida de información**, y el uso de **tarjetas perforadas** como un medio temprano de programación. Aunque estas máquinas no llegaron a terminarse completamente ya que requerían una excesiva cantidad de piezas que no podían producir con la precisión requerida.

Generación 1: Tubos de vacío (1945-1955)

Con el final de la Segunda Guerra Mundial, surgieron las primeras computadoras electrónicas basadas en tubos de vacío o "válvulas". Estas máquinas, como la ENIAC, marcaron un avance revolucionario al ser los **primeros sistemas digitale**s capaces de realizar **cálculos complejos** con una velocidad inédita para la época. Los tubos de vacío, que funcionaban como interruptores electrónicos, tenían importantes limitaciones: generaban gran cantidad de calor, consumían mucha energía y eran propensos a fallos frecuentes.

Las computadoras de esta generación eran de gran tamaño, ocupaban habitaciones enteras y requerían equipos especializados para operar y mantenerlas. Inicialmente, la **programación era compleja, configurándose directamente a través de tableros y conexiones físicas.** Sin embargo, con el tiempo se introdujeron mejoras, como las tarjetas perforadas, que simplificaron la entrada de datos y el control de las instrucciones. Estas máquinas eran principalmente utilizadas en ámbitos militares (por ejemplo, para cálculos balísticos) y científicos (como simulaciones matemáticas), destacando por su enorme capacidad de procesamiento para la época.

Generación 2: Transistores (1955-1965)

La invención del transistor en 1948 revolucionó la computación. Este pequeño dispositivo reemplazó a los tubos de vacío, ofreciendo una alternativa más eficiente, fiable y compacta. Las computadoras a ser considerablemente más pequeñas y rápidas, consumían menos energía y generaban menos calor que sus predecesoras.

Gracias a estas mejoras, las computadoras comenzaron a ser **más accesibles** para empresas y universidades. También se introdujeron **lenguajes de programación de alto nivel**, como FORTRAN y COBOL, **que facilitaron el desarrollo de software y redujeron la dependencia de la programación en lenguaje máquina.**

Esta generación vio la transición de computadoras exclusivamente científicas a aplicaciones comerciales y administrativas, formando el camino para la adopción más generalizada de la tecnología informática.

Generación 3: Circuitos integrados (1965-1980)

La aparición de los circuitos integrados marcó el inicio de esta etapa. **En estos microchips se pudo integrar miles transistores en una sola pieza de silicio**, lo que redujo drásticamente el tamaño de las computadoras, aumentó su eficiencia y capacidad de procesamiento. Esta generación dio lugar a las primeras computadoras verdaderamente compactas, permitiendo que instituciones más pequeñas pudieran adquirirlas.

Además, se desarrollaron los **discos duros** y los **sistemas operativos que permitían multitareas**. Este último fue crucial para reducir costos, ya que varias personas podían usar la misma computadora simultáneamente. En este período, empresas como IBM comenzaron a comercializar computadoras estandarizadas, lo que permitió que el software fuera más compatible entre diferentes modelos.

Generación 4: Computadores personales (1980-presente)

Con el **desarrollo de la tecnología de integración a muy gran escala** (VLSI, por sus siglas en inglés), la cantidad de transistores por chip va creciendo constantemente, dando lugar al microprocesador. Este avance permitió la creación de computadoras personales que eran lo suficientemente pequeñas, económicas y potentes como para ser utilizadas en hogares y pequeñas empresas.

Hubo avances como la creación de la memoria RAM, que hizo posible la aparición de las microcomputadoras, apareció la interfaz gráfica el usuario final, logrando así que fueran más accesibles para todos volviéndose más tarde parte de la vida cotidiana moderna.

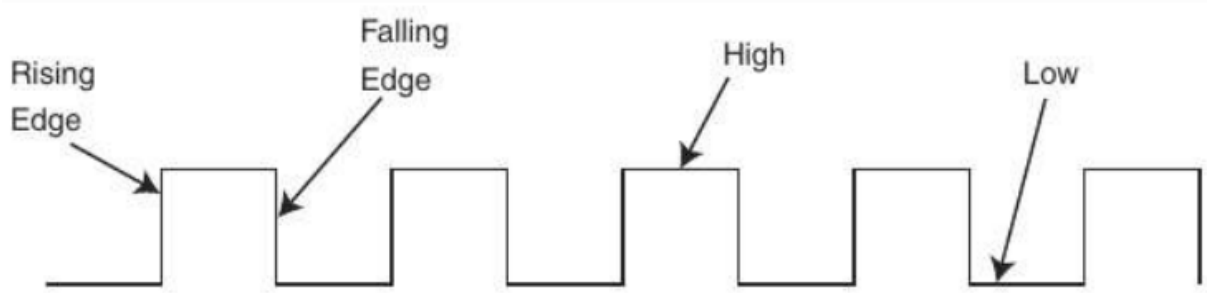
Resumen de avances tecnológicos por generación

Generación	Años	Características
0	hasta 1945	Sistemas mecánicos y electromecánicos
1	1945 a 1954	Tubos al vacío, tableros
2	1955 a 1965	Transistores y sistemas por lotes
3	1965 a 1980	Circuitos integrados
4	desde 1980	VLSI - Computadoras personales y super computadoras

Circuitos y compuertas

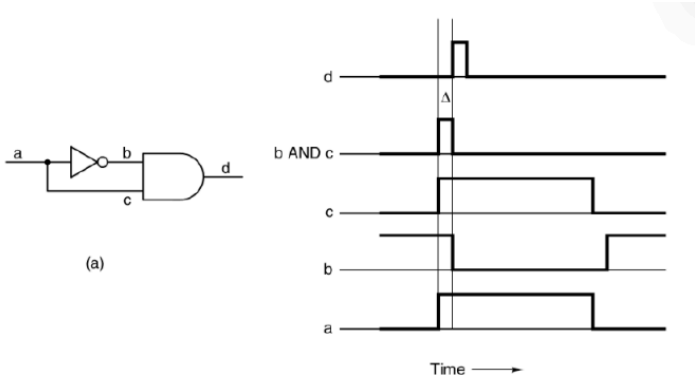
En general, necesitamos una forma de ordenar los diferentes eventos que producen cambios de estados. Para esto usamos relojes: un reloj (clock) es un circuito capaz de producir señales eléctricas oscilantes con una frecuencia uniforme, es decir pulsos. Cada intervalo de tiempo entre dos pulsos consecutivos se conoce como ciclo de tiempo del reloj (clock cycle time)

Las componentes vienen diseñados para detectar un tipo de señal de clock que permiten determinar cómo se mueve de un estado (t) al siguiente (t+1), y así poder sincronizar el circuito. Cuál usar es algo que se elige durante el diseño.
En general, hay dos maneras de medir cambios de fase de las señales:
Cambio de flanco: se detecta cuando hay un cambio en la señal, puede ser flanco ascendente (rising edge) o descendente (falling edge).
Nivel: se verifica que la señal alcance cierto nivel, puede ser alto (high, es decir, un 1) o bajo (low, es decir, un 0).



Para retener sus valores, los circuitos secuenciales recurren a la realimentación (feedback).La realimentación se produce cuando una salida se conecta a una entrada

Flip Flop:



Entrada y Salida (I / O)

Los dispositivos tienen registros específicos que utilizan para comunicarse con la CPU. Estos registros pueden organizarse de dos maneras principales:

- ★ **E/S con puertos dedicados (E/S mapeada por puertos):** Cada registro de control tiene asignado un puerto de E/S. El conjunto de todos los puertos de E/S forman el **I/O port space** que es independiente de la memoria principal. Este espacio está gestionado directamente por la CPU. Para interactuar con los dispositivos en este modelo, se utilizan instrucciones específicas del procesador, como `IN` y `OUT` en arquitecturas x86.
- ★ **E/S mapeada en memoria:**
  Los registros de los dispositivos se les asigna una dirección específica dentro del espacio de memoria principal. Estas direcciones suelen estar reservadas en regiones específicas de la memoria, generalmente en las direcciones bajas o altas, para diferenciarlas del espacio de memoria general. Este método es más simple, ya que utiliza las mismas instrucciones de lectura y escritura que se aplican a la memoria.

Formas de procesar una interrupción:

- Polling:

1. El sistema posee al menos un registro exclusivo para cada dispositivo, la CPU los debe monitorear constantemente.

2. Cuando la CPU detecta que en algún registro hay un dato de relevancia, entonces actúa según corresponda.

   Es el método más lento, pero que tiene una complejidad baja en cuanto al hardware. *El Módulo de E/S se conecta solo con la Memoria, no tiene conexión con la CPU.*

- Interrupciones

1. El dispositivo periférico genera una petición de interrupción, cuando está listo o necesita atención, a la línea de interrupción.

2. La CPU finaliza la instrucción en curso, y verifica si hay señales de interrupción activas.

3. Si se detecta el pedido de interrupción, deshabilita temporalmente las interrupciones. Este paso es fundamental para mantener la estabilidad del sistema ya que pueden existir múltiples dispositivos solicitando interrupción al mismo tiempo.

4. La CPU almacena el contexto actual del programa en la pila de memoria. Como contexto englobamos el PSW (Program Status Word), que contiene información sobre el estado actual del procesador como las flags de condición, el PC (Program Counter), que indica la dirección de la próxima instrucción a ejecutar y registros de propósito general.

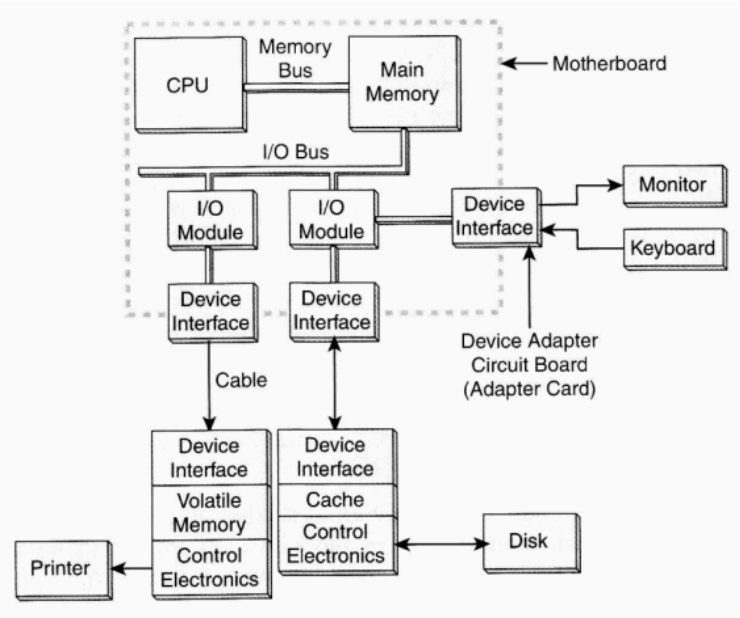
5. Detecta quién envió la señal. Para decidir el orden de prioridades que va a atender, hay dos procesos de reconocimiento:

   Auto Vectorización (soft): La CPU ejecuta un software que interroga al controlador o a los dispositivos para identificar al emisor de la interrupción.
   Vectores de interrupción (hard): Se utiliza una tabla predefinida en el sistema donde cada línea de interrupción está asociada con una dirección específica de memoria, que apunta directamente a la rutina que atenderá la interrupción.

6. Ejecuta la rutina de atención (ISR) :

7. Restaura el contexto para que la CPU continúe con su trabajo anterior y habilita nuevamente las interrupciones.

- **DMA (Direct Memory Access)**



El **DMA** es un método avanzado en el que un controlador especial (controlador DMA) gestiona directamente la transferencia de datos entre un dispositivo periférico y la memoria principal, sin la intervención constante de la CPU. Este controlador comparte el mismo BUS de datos con el que la CPU se comunica con la Memoria. De esta forma la CPU le deja el trabajo al controlador y no interviene durante la transferencia de datos.

- Caso para grandes volúmenes de datos.
- Requiere el hardware adicional

Buses

La **CPU** necesita comunicarse con otros componentes del sistema para coordinar y realizar las operaciones necesarias. Para esto, se utiliza el **bus**, un conjunto de líneas o cables que permiten la transferencia de datos, direcciones y señales de control entre los distintos componentes del sistema. Los buses trabajan con datos en paralelo, lo que significa que los bits se transfieren simultáneamente a través de múltiples líneas.

El diseño del bus permite conectar varios dispositivos al sistema, facilitando la expansión. Sin embargo, compartir un bus común tiene desventajas: solo un dispositivo puede usar el bus a la vez, lo que genera un **cuello de botella** en la comunicación. Además, la velocidad del bus puede verse afectada por la distancia entre los componentes, ya que los cables más largos introducen retrasos.

Para organizar los dispositivos conectados al bus, se los puede clasifica en:

Master: dispositivos que inician las operaciones

Slave: dispositivos que responden a las solicitudes del Máster

Aunque podemos separar los buses en distintas clasificaciones, según características:

Según forma en que están conectadas:

- Los buses puede ser "punto por punto", conectando específicamente dos componentes
- Pueden ser un camino de datos común el cual conecta varios dispositivos, los cuales requieren compartirlo.

Según función:

Data Bus - Bus de datos:

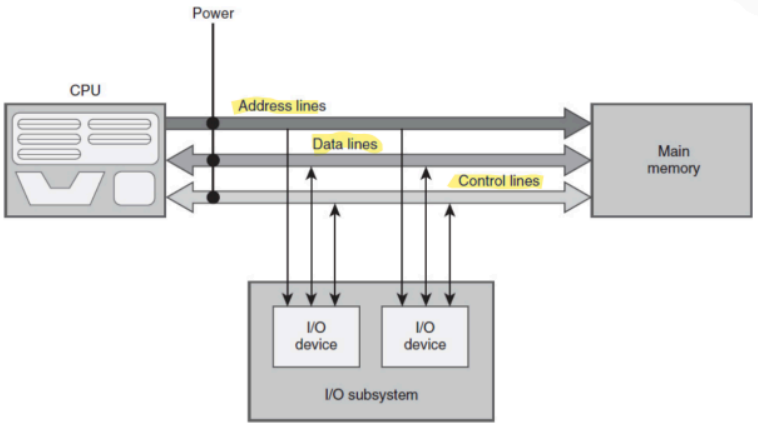
- Transporta los datos que se están transfiriendo entre los componentes.

Address Bus - Bus de direcciones:

- Transporta las direcciones de memoria o las identificaciones de dispositivos a los que se accede.

Control Bus - Bus de control:

- Indican cual es el dispositivo que tiene permiso para utilizar el bus y para que lo va a utilizar



Según su sincronización:

- **Bus Sincrónico:** Está gobernado por un reloj común. Todos los eventos ocurren en sincronía con los ticks del reloj, lo que garantiza que las transferencias se completen en un tiempo predecible. Este diseño es más simple y eficiente, pero depende de que todos los dispositivos puedan operar a la misma velocidad.

Dos ticks de reloj

- **Bus Asincrónico:**
  En este tipo de bus, las operaciones no dependen de un reloj central. En su lugar, las líneas de control y un protocolo de **handshaking** (apretón de manos) coordinan la transferencia de datos. En este protocolo, el dispositivo que desea comunicarse envía una señal de solicitud (request) indicando que está listo para enviar o recibir datos. El procesador o dispositivo receptor responde con una señal de reconocimiento (acknowledge), y solo entonces se realiza la transferencia de datos. Este enfoque es más flexible, ya que permite que los dispositivos trabajen a su propio ritmo, pero implica una mayor complejidad y posibles retrasos en la comunicación debido a la coordinación necesaria entre las partes.

Tiempo necesario

Ciclo del Bus:

Cada transferencia de datos en el bus ocurre durante un ciclo de bus, que es el tiempo entre **dos ticks del reloj** (en buses sincrónicos) o **el tiempo necesario para completar un protocolo de sincronización** (en buses asincrónicos). Este ciclo incluye la preparación de los datos, el acceso a las direcciones correspondientes, y la comunicación de señales de control.

En la arquitectura de Von Neumann:

- Existe un único bus que transporta tanto instrucciones como datos entre la CPU, la memoria y los dispositivos de E/S

Memoria

Son dispositivos utilizado para el almacenamiento de información
La memoria es un componente fundamental en las computadoras, ya que permite almacenar y recuperar datos e instrucciones que el procesador necesita para funcionar. Puede clasificarse de diferentes maneras según sus características, por ejemplo podemos distinguirlas principalmente por:

- **Volátiles**: son aquellas que requieren energía para mantener la información almacenada; conserva sus contenidos mientras está encendida y cuando se apaga los datos se pierden inmediatamente o muy rápidamente.
    - **Memoria caché**: Es una memoria pequeña y rápida (pero más cara) generalmente sirve para tener a más accesibles los datos que se están utilizando más frecuentemente.
    - **RAM** (random-access memory): La cual en realidad es una memoria que se puede leer y escribir, es usada para guardar los datos y los programas que está por ejecutar la computadora, esta memoria es volátil lo que quiere decir que cuando no tenga energía todo lo almacenado se borra
    - **Memorias SRAM** (Static Random Access Memory)

- **No Volátiles**: Aquellas que pueden recuperar la información almacenada, incluso después de haber sido desconectadas.
    - **ROM:** tienen guardada información crítica, funcionamiento básico del sistema, para el funcionamiento del sistema, esta memoria no es volátil por lo que la información guardada en ella no se borrará aunque no haya energía. Esta memoria se utiliza en sistemas donde no es necesario cambiar la programación
      Ejemplos de tipos de ROM: PROM, EPROM y EEPROM

Jerarquía de memoria:

Como no todas las memorias son igual de rápidas, baratas y eficientes las computadoras utilizan combinaciones de ellas intentando obtener el mejor rendimiento al menor costo posible para eso existe la jerarquía, la cual es una regla: cuanto más rápida sea la memoria más caro será cada bit guardado en ella.

Las memorias que constituyen la jerarquía normalmente son:

- **Registros:** Son pequeños bloques de memoria integrados directamente en la CPU.
    - Velocidad extremadamente alta, ya que están dentro del procesador.
    - Almacenan temporalmente los datos más inmediatos necesarios para las operaciones en curso.

Usados principalmente para cálculos aritméticos y direccionamiento de memoria.

- **Caché:** Es una memoria rápida donde se guardan temporalmente los datos usados con frecuencia, está conectada a una memoria principal.

  Almacenar copias de los datos más frecuentemente usados de la RAM para un acceso rápido. Reduce drásticamente el tiempo de espera del procesador al acceder a datos recurrentes.

- **Memoria principal (RAM) :** Es la memoria principal que la CPU utiliza para almacenar datos y programas activos. Proporcionar acceso rápido a datos que no caben en la caché.
    - Es volátil
    - Más lenta que el caché pero con mayor capacidad

- **Memoria secundaria:** Generalmente se compone de un disco duro (por ejemplo un HDD) el cual contiene datos que no es accesible de forma directa para el CPU por lo cual estos datos necesitan pasar por la memoria principal para ser accedidos

  Almacenar grandes cantidades de datos que no necesitan estar inmediatamente disponibles para la CPU.

- **Otras**: son aquellos que requieren una intervención fuera del sistema para ser conectados (por ejemplo: un pendrive), los datos para ser accesibles necesitaran ser transferidos a la memoria secundaria

Caché

Se trata de una memoria de alta velocidad y capacidad limitada que actúa como intermediaria entre la Unidad Central de Procesamiento (CPU) y la memoria principal (RAM). Su objetivo principal es reducir el tiempo que la CPU necesita para acceder a los datos y las instrucciones que utiliza con mayor frecuencia.

La memoria caché almacena temporalmente una copia de los datos e instrucciones que la CPU utiliza o necesitará pronto. La CPU puede acceder a estos datos mucho más rápido desde la caché que desde la RAM, lo que acelera las operaciones. Este diseño se basa en dos principios fundamentales de los patrones de acceso a datos:

- **Localidad temporal**: Los datos que se han utilizado recientemente tienen una alta probabilidad de volver a ser usados en el corto plazo.
- **Localidad espacial**: Si se accede a una dirección de memoria, es probable que las direcciones cercanas también sean necesarias.

Dado que el acceso a la memoria principal es significativamente más lento que el acceso a la memoria caché, ésta ayuda a minimizar los retrasos que la CPU enfrentaría si tuviera que depender únicamente de la RAM para obtener datos.

Jerarquía de la memoria caché

La memoria caché se organiza en niveles jerárquicos, cada uno con diferentes tamaños, velocidades y ubicaciones:

- **Caché L1**: Es la más pequeña y rápida, ubicada dentro del núcleo del procesador. Suele estar dividida en dos partes: una para datos y otra para instrucciones.
- **Caché L2**: Es más grande que L1 pero ligeramente más lenta. También está ubicada cerca del núcleo de la CPU y sirve como respaldo en caso de que los datos no se encuentren en L1.
- **Caché L3**: Es compartida entre todos los núcleos de la CPU. Tiene una capacidad mayor, pero su velocidad es menor comparada con L1 y L2.

La CPU primero busca los datos necesarios en la caché L1. Si no los encuentra, consulta L2 y, finalmente, L3 antes de acceder a la memoria principal.

Funcionamiento de la memoria caché

Cuando la CPU necesita un dato o instrucción, se inicia un proceso llamado **búsqueda en la caché**. Si el dato está presente en la caché, ocurre un "acierto de caché", y la CPU lo utiliza directamente. Si no está, ocurre un "fallo de caché", y la CPU debe buscarlo en niveles inferiores o en la RAM, lo que implica mayor tiempo de espera.

Para gestionar eficientemente qué datos se almacenan o reemplazan en la caché, se aplican diferentes **políticas de reemplazo**:

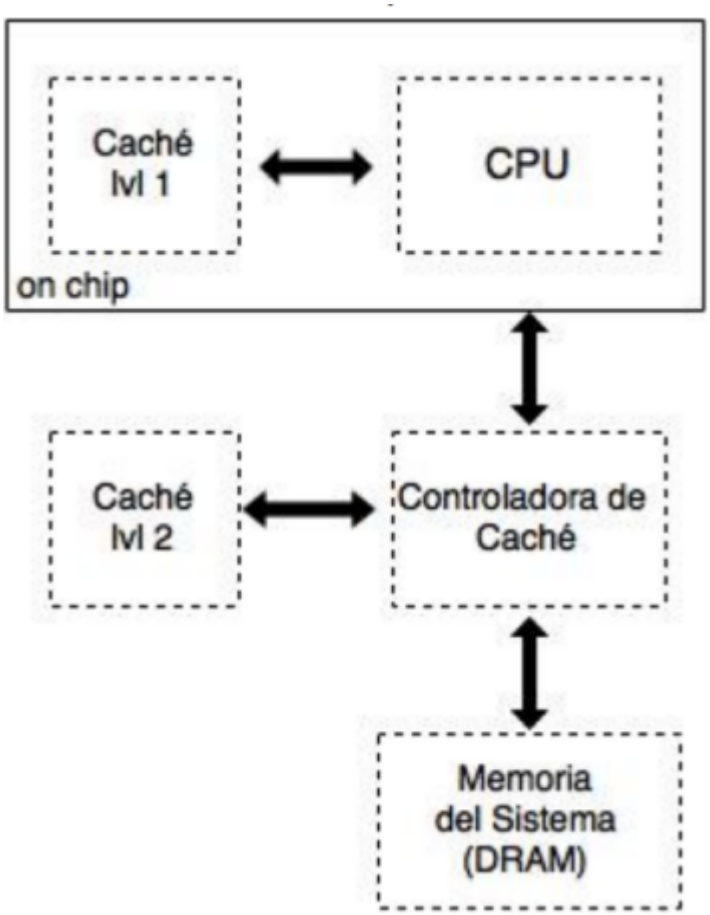
- **FIFO (First In, First Out)**: Los datos más antiguos son reemplazados primero.
- **LRU (Least Recently Used)**: Los datos menos utilizados recientemente son eliminados.
- **Random**: Los datos a reemplazar se seleccionan de manera aleatoria, aunque es menos común.

Además, existen estrategias para manejar la escritura en caché:

- **Write-through**: Cada vez que se escribe un dato en la caché, también se actualiza en la memoria principal.
- **Write-back**: Los datos modificados en la caché solo se escriben en la memoria principal cuando se eliminan de la caché.

Beneficios de la memoria caché

La principal ventaja de la memoria caché es la **reducción de la latencia** en el acceso a datos. Al guardar los datos más utilizados cerca de la CPU, se minimizan los tiempos de espera, lo que aumenta el rendimiento del sistema. Además, al reducir el número de accesos a la RAM, se optimiza el uso de la memoria principal y se libera ancho de banda para otras operaciones.



Definiciones importantes

Estructura: la forma en que las componentes están interrelacionadas para llevar a cabo una tarea.

Función: la operación de cada componente individual como parte de la estructura (procesamiento, almacenamiento, transferencia de datos, control).

Mainframes: Se refieren a grandes computadoras centrales en los años 50´ que requieren instalaciones especiales para su funcionamiento debido a su tamaño, consumo energético y necesidades de enfriamiento.

PC (Program Counter): indica la dirección de la próxima instrucción a ejecutar.

PSW (Program Status Word), que contiene:

- Flags de condición (por ejemplo, si ocurrió un carry, overflow, resultado cero, etc.).
- Modos de operación de la CPU.
- Indicadores de habilitación/deshabilitación de interrupciones.

Rutina de Atención de Interrupciones (ISR):
Es un conjunto de instrucciones específicas diseñadas para manejar un evento de interrupción. Es un pequeño programa que se ejecuta en lugar del programa principal hasta que la interrupción ha sido atendida.

ISA - Instruction Set Architecture:
La ISA es el set de instrucciones de la arquitectura. Es la interfaz que existe entre todo el software que corre en la máquina y el hardware que lo ejecuta. Nos permite “hablarle” a la máquina. Está íntimamente relacionada con la organización del sistema, pues basa su implementación en términos de componentes electrónicos.
Define la forma en la cual un programador observa la arquitectura del sistema.

Principio de equivalencia Hardware-Software: “Cualquier cosa que puede ser hecha por software puede ser hecha en hardware y viceversa”

Ley de Moore: La densidad de transistores en un circuito integrado se duplica cada año. La versión actual de la ley dice que la densidad de los chips de silicio se duplica cada 18 meses. Esta ley no puede mantenerse para siempre, por limitaciones físicas.

Ley de Rock: A medida que aumenta la complejidad de los circuitos, el costo de fabricación por transistor disminuye de manera exponencial. Sin embargo, el costo total del diseño y fabricación del chip tiende a incrementarse significativamente.

Ley de Amdahl: Cuando mejoras una parte de un sistema, el beneficio total que obtienes depende de cuánto tiempo dedica el sistema a esa parte mejorada. En otras palabras, incluso si haces una parte del trabajo mucho más rápida, el impacto en el rendimiento total

será limitado por el tiempo que el sistema pasa haciendo el resto del trabajo que no se puede mejorar.

- Partes secuenciales; partes que requieren un orden específico y entonces no se puede dividir porque necesita respetar el orden.

Computación paralela:
Se refiere a un enfoque en la programación y diseño de sistemas informáticos en el cual múltiples tareas o procesos se ejecutan simultáneamente para resolver un problema o realizar una operación de manera más rápida y eficiente. En lugar de realizar una tarea tras otra de forma secuencial (una tras otra), se divide el trabajo en partes independientes que pueden realizarse al mismo tiempo en diferentes procesadores o núcleos.

- Las tareas paralelas suelen necesitar coordinarse entre sí. Este proceso de sincronización puede causar **cuellos de botella**, ya que algunas tareas deben esperar a que otras terminen.
- Dividir las tareas y coordinar su ejecución requiere **procesos adicionales** que pueden consumir recursos.

Overflow:

Ocurre cuando el resultado de una operación aritmética excede el rango que puede representarse con el número de bits disponibles en el sistema. En otras palabras, el valor calculado es demasiado grande o demasiado pequeño para ser almacenado correctamente en el espacio reservado.

Sistema embebido:

Es un sistema computacional diseñado para realizar tareas específicas dentro de un dispositivo más grande. A diferencia de una computadora general, que puede ejecutar múltiples aplicaciones para propósitos generales, un sistema embebido está integrado en un producto o máquina para cumplir una función particular.

Retroalimentación: Es decir, garantizan que, a partir de la ISA que funciona como frontera que nos separa del HW, el procesador sigue aceptando las instrucciones de procesadores anteriores, permitiendo ejecutar código antiguo sobre una organización moderna.

Rutina de Atención de Interrupciones: subrutina que contiene las instrucciones necesarias para atender la interrupción generada por un dispositivo o evento específico. Engloba también las instrucciones asociadas al pedido de interrupción del dispositivo.

Flip Flop:

- **Flip-Flop SR:** Uso básico de memoria.
- **Flip-Flop D:** Sincronización y almacenamiento.
- **Flip-Flop JK:** Circuitos secuenciales, contadores.
- **Flip-Flop T:** Divisores de frecuencia y contadores binarios.
- **Flip-Flop Maestro-Esclavo:** Aplicaciones sincronizadas en sistemas digitales.

Cada flip-flop tiene aplicaciones específicas según las necesidades de diseño de sistemas digitales