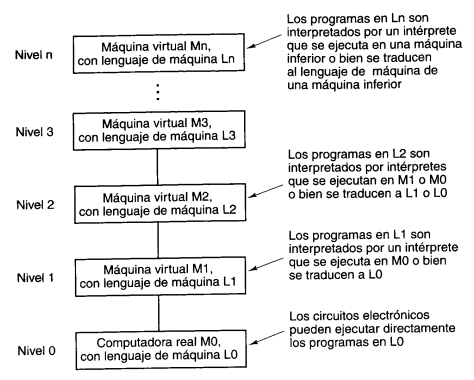
1. **Describa el concepto de capas de abstracción por máquinas virtuales y detalle por qué es beneficioso generar dichas capas entre el lenguaje de máquina y el lenguaje humano. ¿Cuántas máquinas se pueden crear sobre una máquina M0?**
   * El concepto de capas de abstracción por máquinas virtuales se refiere a la estrategia de diseñar conjuntos de instrucciones más accesibles para los humanos que el conjunto de instrucciones de máquina original. Esto se logra mediante la creación de un lenguaje de alto nivel (LI) que se traduce o interpreta en el lenguaje de máquina original (LO) antes de su ejecución en la computadora.
   * La traducción implica convertir un programa escrito en LI en un equivalente en LO antes de ejecutarlo, mientras que la interpretación implica ejecutar directamente un programa escrito en LI mediante un intérprete que lo analiza e implementa instrucción por instrucción.
   * Ambos métodos permiten que los programadores escriban en un lenguaje más humano y accesible, lo que facilita el desarrollo de software y la comprensión del funcionamiento interno de la computadora.
   * Para mejorar aún más la eficiencia y la facilidad de uso, se pueden crear nuevos conjuntos de instrucciones (como L2) que estén más orientados hacia las personas y menos hacia la máquina. Esto crea una serie de capas de abstracción, cada una más accesible y sofisticada que la anterior.

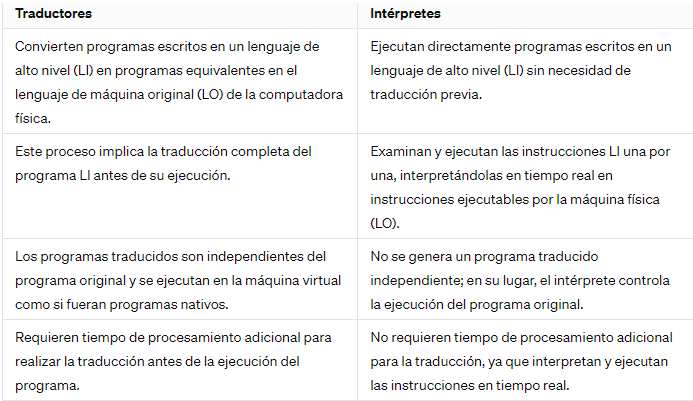
En resumen, las capas de abstracción por máquinas virtuales permiten que los programadores escriban programas en un lenguaje más accesible, facilitando así el desarrollo de software y la interacción con la computadora.



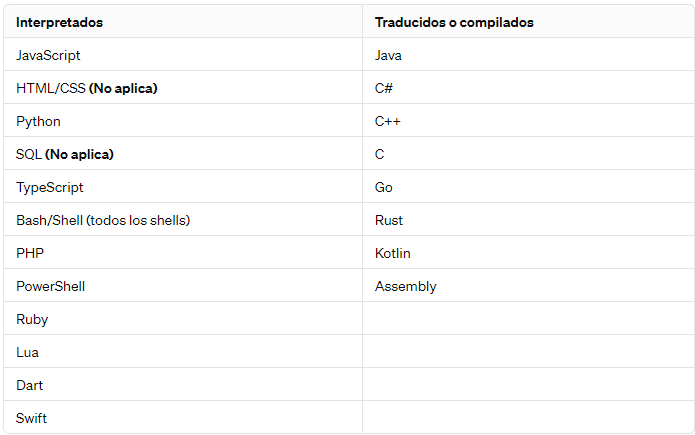
En un caso ideal, considerando que no hay limitaciones físicas y que la máquina M0 tiene recursos ilimitados, teóricamente se podrían crear un número infinito de máquinas virtuales sobre esta máquina. Esto implicaría que no habría restricciones en términos de capacidad de procesamiento, memoria, almacenamiento o ancho de banda de red.

Para un caso real la cantidad exacta de máquinas virtuales que se pueden crear sobre una máquina M0 dependerá de la capacidad y la configuración específica de esa máquina, así como de los requisitos individuales de cada máquina virtual.

1. **Defina que son los traductores e intérpretes. Tabule sus diferencias. Entre en la siguiente página:** [**https://survey.stackoverflow.co/2023/#technology**](https://survey.stackoverflow.co/2023/%23technology%20) **y realice una tabla que lista cuáles de los primeros 20 lenguajes de programación más populares son interpretados y cuáles son traducidos.**
   * **Traductores**: son programas que convierten programas escritos en un lenguaje de alto nivel (LI) en programas equivalentes en el lenguaje de máquina original (LO) de la computadora física. Este proceso implica la traducción completa del programa LI antes de su ejecución. Los programas traducidos son independientes del programa original y se ejecutan en la máquina virtual como si fueran programas nativos.
   * **Intérpretes**: son programas que ejecutan directamente programas escritos en un lenguaje de alto nivel (LI) sin necesidad de traducción previa. Estos programas examinan y ejecutan las instrucciones LI una por una, interpretándolas en tiempo real en instrucciones ejecutables por la máquina física (LO). No se genera un programa traducido independiente; en su lugar, el intérprete controla la ejecución del programa original.



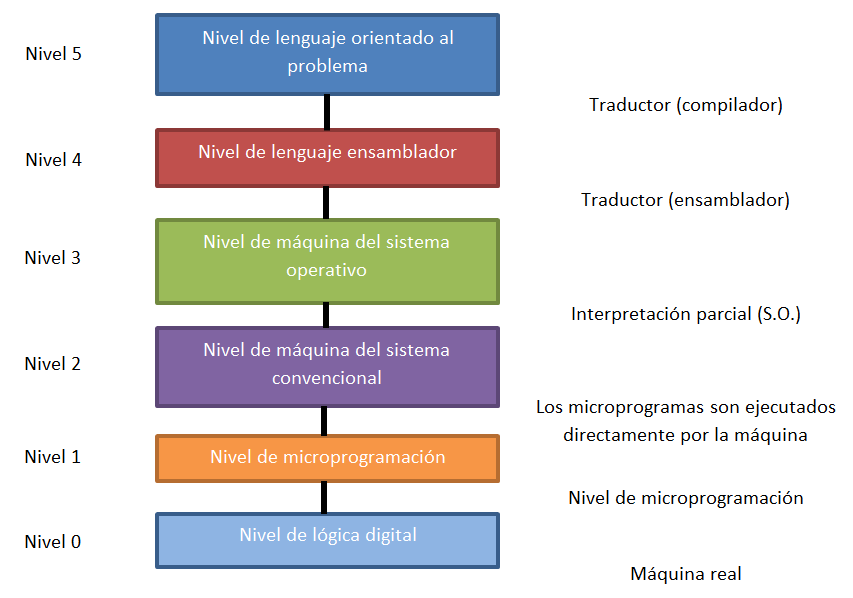
**Tabla de Lenguajes:**



**HTML/CSS (No aplica):** HTML y CSS son lenguajes de marcado y estilo utilizados en el desarrollo web para estructurar y dar estilo a páginas web. No son lenguajes de programación en el sentido tradicional y no se clasifican como interpretados o traducidos, ya que se interpretan directamente por los navegadores web para renderizar páginas.

**SQL (No aplica):** SQL (Structured Query Language) es un lenguaje de consulta utilizado para interactuar con bases de datos relacionales. Aunque se ejecuta en motores de bases de datos, no se interpreta o traduce como los lenguajes de programación convencionales. En lugar de eso, se interpreta directamente por el motor de base de datos para realizar operaciones en la base de datos.

1. **Diagrame la máquina multinivel contemporánea, detalle cada nivel y como es la comunicación entre cada uno.**



**DETALLE POR NIVEL:**

**Nivel 5: Nivel de Lenguaje Orientado al Problema:**

En este nivel se encuentran los lenguajes de programación de alto nivel, diseñados para resolver problemas específicos. Los programas escritos en estos lenguajes se traducen o interpretan en niveles inferiores antes de ser ejecutados.

**Nivel 4: Nivel de Lenguaje Ensamblador:**

Aquí se utiliza un lenguaje simbólico más comprensible que el lenguaje de máquina para programar la computadora. Los programas escritos en lenguaje ensamblador se traducen a lenguaje de máquina antes de ser ejecutados.

**Nivel 3: Nivel de Máquina del Sistema Operativo:**

Este nivel gestiona las operaciones del sistema operativo, como el manejo de memoria, la planificación de procesos y la gestión de dispositivos. Puede interpretar algunas instrucciones y ejecutar otras directamente mediante el hardware.

**Nivel 2: Nivel de Máquina del Sistema Convencional:**

Aquí se encuentra la micro arquitectura de la computadora, que incluye registros y una ALU (Unidad de Aritmética Lógica). Los registros almacenan datos y la ALU realiza operaciones aritméticas y lógicas.

**Nivel 1: Nivel de Microprogramación:**

Este nivel controla directamente la operación de la lógica digital. Define la operación de la máquina a un nivel muy bajo, incluyendo cómo se realizan las operaciones lógicas y aritméticas básicas.

**Nivel 0: Nivel de Lógica Digital:**

En este nivel, se encuentran las compuertas lógicas que se construyen con transistores. Estas compuertas realizan operaciones lógicas básicas como AND, OR, etc. Pueden combinarse para formar estructuras más complejas como memorias y circuitos de procesamiento.

**COMUNICACIÓN ENTRE NIVELES:**

**Nivel 5 - Lenguaje Orientado al Problema:**

En este nivel, los programas están escritos en lenguajes de alto nivel orientados a resolver problemas específicos. La comunicación se realiza a través de compiladores o intérpretes, que traducen o interpretan estos programas en instrucciones ejecutables por el hardware de la computadora.

**Nivel 4 - Lenguaje Ensamblador**:

La comunicación en este nivel se lleva a cabo mediante programas escritos en lenguaje ensamblador, que se traducen en instrucciones de nivel inferior entendidas por el hardware de la computadora.

**Nivel 3 - Máquina del Sistema Operativo:**

En este nivel, la comunicación se realiza a través de las instrucciones del sistema operativo, que coordinan el acceso a los recursos del sistema y administran la ejecución de los programas de usuario.

**Nivel 2 - Máquina del Sistema Convencional:**

La comunicación en este nivel implica la ejecución de instrucciones de bajo nivel que controlan directamente la unidad central de procesamiento (CPU) y otros componentes del hardware de la computadora.

**Nivel 1 - Microprogramación:**

En este nivel, la comunicación se realiza mediante la definición y ejecución de microinstrucciones que controlan las operaciones fundamentales del hardware de la computadora, como la ejecución de operaciones aritméticas y lógicas.

**Nivel 0 - Lógica Digital:**

Este nivel se comunica directamente con los componentes físicos del hardware de la computadora, como las compuertas lógicas, a través de señales eléctricas digitales.

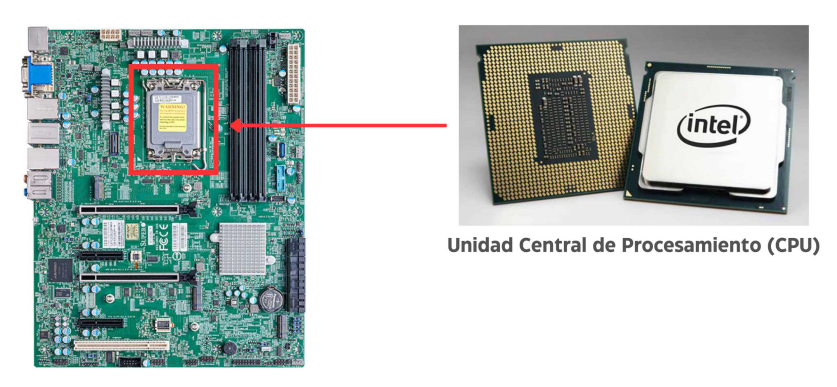
1. **¿Cuáles son los elementos fundamentales de una computadora? Describa brevemente el propósito de cada uno**

Toda computadora tiene como mínimo tres conceptos fundamentales:

* **Procesador:** Es el “cerebro” de la computadora y su propósito es ejecutar las instrucciones presentes en un programa. Mejor conocido como unidad central de procesamiento (CPU).

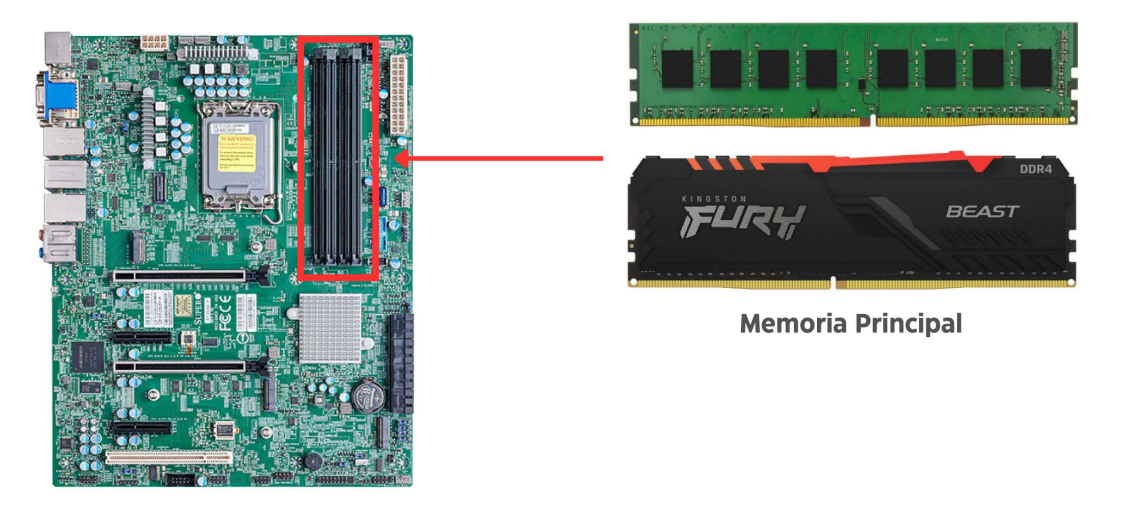
El ciclo típico para ejecutar una instrucción es:

* 1. Busca la instrucción de memoria.
  2. La examina
  3. La ejecuta

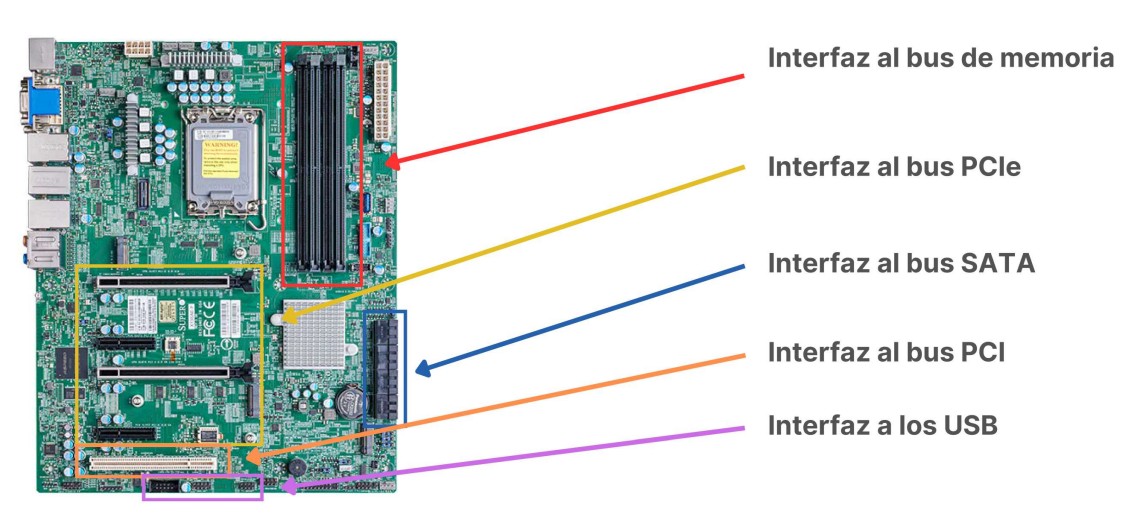
Y repite hasta terminar su ejecución. 

* **Memoria Principal:** es la parte de la computadora donde se cargan los programas y sus datos.

Consiste de celdas, a donde cada una puede guardar un poco de información( por lo generar 8 bits o 1 Byte). Cada una también tiene una dirección en memoria, en otras palabras, adonde entre todas las celdas se la puede encontrar.

****

* **Bus:** El bus es un canal de comunicación que permite que la transferencia de datos entre todos los componentes de una computadora. Una computadora puede tener uno o más.

****

Aunque lo fundamental es el procesador, la memoria y el bus, hay otros tipos de componentes, conocidos como periferios, que son útiles para el funcionamiento de una computadora.

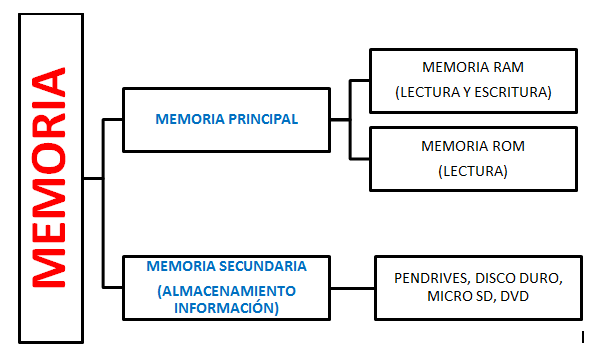


1. **¿Qué diferencia hay cuando nos referimos a memoria y almacenamiento? ¿Cuáles de estos es la memoria principal y la memoria secundaria?**

La memoria es la parte de la computadora en la que se almacenan programas y datos.

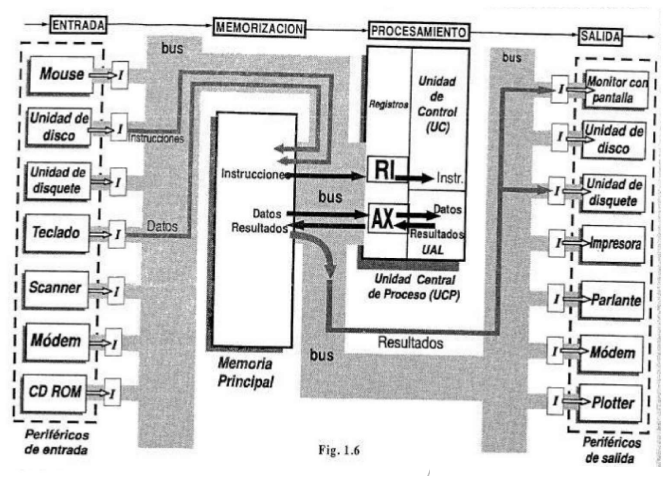
Almacenamiento es utilizada para guardan información debido a que la memoria es demasiado pequeña para contener demasiada información.

Conocemos a la memoria como Memoria principal y al Almacenamiento como Memoria secundaria.



1. **Explique en detalle la analogía del modelo inteligente.**

Consta de un proceso que se puede dividir en cuatro etapas:

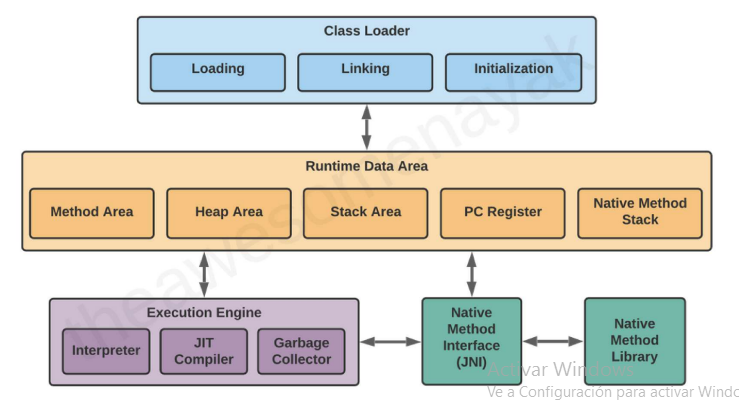
* Entrada
* Memorización
* Procesamiento
* Salida

La entrada nos entrega algún tipo de dato. Consecuentemente esta se comunica con la memoria principal atravez del bus. Una vez identificado el dato a procesar, este es llevado a la unidad de procesamiento (procesador) donde se procesa y el resultado es almacenado para ser utilizado cuando sea requerido. Una vez solicitado es transportado atravez del bus desde la unidad de memoria hasta la unidad de salida solicitado ( al periférico solicitado ).

1. **Mencione los elementos principales de la Arquitectura de la JVM (Java Virtual Machine) y describa brevemente cada uno.**

Arquitectura de la JVM:

* Class Loader
* Runtime Memory/Data Area
* Execution Engine



**Class Loader;**

* **Loading:** Está a cargo de cargar el código compilado a bytecode a la memoria principal para que la pueda ser utilizado. Hace la carga en tres partes primero carga las clases estándar de Java, segundo carga extensiones de las clases estándar, y finalmente carga las clases presente en la aplicación
* **Linking:** Después que las clases son cargadas a memoria, tienen que ser linkeadas.
* **Initialization:** Involucra ejecutar el método inicial de la clase principal

**Runtime Data Area**

* **Method Area:** Contiene las constantes y atributos definidos en las clases y también el código para los constructores y métodos
* **Heap Area:** Todo objeto o dato instanciado en tiempo de ejecución están aquí.
* **Stack Area:** Las variables locales, llamados a métodos y resultados parciales son guardados aquí.
* **Program Counter (PC) Registers:** Mantiene la dirección en memoria de la instrucción actual que se está ejecutando en la JVM
* **Native Method Stacks:** Similar al Stack Area pero para métodos que estén escritos en otro lenguaje aparte de Java.

**Ececution Engine:**

* **Interpreter:** Ejecuta línea por línea el bytecodedel programa de forma interpretada.
* **JIT Compiler:** Este ayuda con la desventaja del intérprete lento. El mismo compila el bytecode a código de maquina donde llamadas consecutivas al mismo código ejecuta directamente el código de máquina, mejorando el rendimiento.
* **Garbage Collector**: automáticamente remueve objetos que no son referenciados del Heap Area, liberando memoria.