Organización estructurada de computadoras

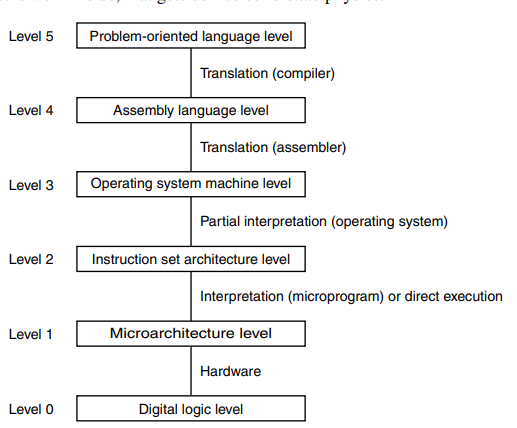
1. Si se puede resumir el concepto de capas de abstracción por máquinas virtuales, podemos decir simplemente que se trata de niveles que separan el funcionamiento más técnico de un VM, de la experiencia del usuario y su facilidad para resolver su necesidad. Si hablamos más a profundidad, decimos que existen distintos niveles de máquinas virtuales a las que podemos llamar M*n*siendo *n* un número indeterminado empezando por 0, que es el nivel más bajo. Todas acompañadas de su correspondiente lenguaje L*n* para la situación específica. El nivel 0 es el más cercano a los componentes electrónicos, una máquina M0 utiliza L0 para comunicarse directamente con estos componentes con *built-in* instrucciones, mientras que por cada nivel que se suba, se acerca más al lenguaje humano, creando nuevas instrucciones traducidas o interpretadas a una máquina inferior. Esto nos beneficia ya que puede llegar a ser más cómodo y rápido sin tener que preocuparse por los *n* niveles subyacentes al que está manejando.
2. Un traductor es, como dice su nombre, un proceso que traduce el código que escribimos una vez en el n-level que usamos a uno más bajo, usando las instrucciones existentes en L0 necesarias para hacer una equivalencia a nuestro código de n-level. Una vez hecho esto, deja de lado nuestro código L*n* y ejecuta el nuevo programa L0 que controla la computadora.

Un intérprete, por otro lado, en lugar de crear un nuevo programa, usa el L*n* código que creamos como datos de entrada examinando cada instrucción y ejecutando un equivalente en L0 sin tener que crear un nuevo programa.

**­­Top 20 lenguajes más populares.**

|  |  |
| --- | --- |
| **Nombre** | **Tipo** |
| 1. **Javascript** | **Interpretado** |
| 1. **Python** | **Interpretado** |
| 1. **SQL** | **Interpretado (digamos)** |
| 1. **Typescript** | **Interpretado y Traducido** |
| 1. **Bash/Shell** | **Interpretado** |
| 1. **Java** | **Interpretado y Traducido** |
| 1. **C#** | **Interpretado y Traducido** |
| 1. **C++** | **Interpretado y Traducido** |
| 1. **C** | **Traducido** |
| 1. **PHP** | **Interpretado y Traducido** |

|  |  |
| --- | --- |
| 1. **PowerShell** | **Interpretado y Traducido** |
| 1. **Go** | **Traducido** |
| 1. **Rust** | **Traducido** |
| 1. **Kotlin** | **Interpretado y Traducido** |
| 1. **Ruby** | **Interpretado** |
| 1. **Lua** | **Interpretado** |
| 1. **Dart** | **Interpretado y Traducido** |
| 1. **Assembly** | **Traducido** |
| 1. **Swift** | **Traducido** |
| 1. **R** | **Interpretado** |

****

Para entender este diagrama, lo voy a explicar visto desde abajo hasta arriba. O sea, voy a empezar con el **nivel 0, el nivel de lógica digital:**

En este nivel se encuentran lo que llamamos **compuertas lógicas**, o simplemente **gates**. Estas compuertas están hechas de componentes analógicos como transistores y puede almacenar señales representadas como 0 (*false*) o 1 (*true*) dando una salida combinada AND, OR, NOR, etc.

Un circuito está formado por una combinación de compuertas que forman una memoria de 1-bit y en grupos de 16, 32 o 64 conformando un registro.

El **nivel 1, nivel de microarquitectura** es el que usa estos registros del nivel 0 para formar una memoria local de 8 a 32 registros y un circuito llamado **ALU (Unidad de lógica Aritmética)** la cual puede realizar estas operaciones simples. Está conectado a los registros para formar un **data path** para que fluyan los datos, seleccionando uno o dos y guardando los resultados en algunos registros.

También, estas operaciones son controladas directamente por el Hardware, en algunas máquinas completamente y en otras parcialmente y con ayuda de un programa llamado **microprograma** el cual es un intérprete L2.

El **nivel 2, ISA level o Nivel de set de instrucciones de arquitectura** se puede decir que es el que lleva los manuales y describe las instrucciones que interpreta el microprograma o el hardware que ejecuta los circuitos.

El **nivel 3, sistemas operativos**, es un nivel que maneja instrucciones L3. Se puede decir que es un set de nuevas instrucciones, una diferente organización de memoria y la habilidad de correr 2 o más programas a la vez con otro tipo de mejoras. Pero algunas de estas operaciones son interpretadas por el sistema operativo mientras que otras son idénticas a L2 siendo ejecutadas directamente por el microprograma o el hardware.

El **nivel 4, nivel del Ensamblador**, es el nivel que empieza a acercarse más al lenguaje humano. Para poner en contexto, los anteriores lenguajes consisten en series de números grandes, que el humano corriente difícilmente entenderá. A partir de Assembly, se empiezan a usar palabras y abreviaciones el cual traduce al nivel necesario para que la máquina pueda interpretarlo.

Finalmente, en el **nivel 5, lenguajes de alto nivel**, se encuentran los lenguajes de programación que usa cualquier programador de aplicación para resolver problemas. Estos facilitan todo y son usualmente interpretados o bien traducidos a nivel 3 o 4. Aquí se encuentran C, C++, PHP, etc.

1. Los elementos fundamentales de una computadora son:

* **CPU:** Se puede definir como el cerebro de la computadora, es el que ejecuta las instrucciones dadas y procesa todos los datos, a la vez que coordina lo que hacen el resto de componentes.
* **Memoria:** Aquí englobo tanto RAM como los discos HDD o SSD. La RAM es la encargada de los procesos en ejecución mientras que los discos almacenan los datos de forma permanente.
* **Motherboard:** Es un circuito que conecta todos los componentes del hardware logrando su comunicación y transferencia de datos.
* **Fuente de alimentación:** Esta brinda electricidad a cada componente de la computadora regulándola según las necesidades de cada uno para el funcionamiento correcto.
* **Periféricos y dispositivos de E/S:** Aquí existen las impresoras, los monitores, los teclados, la salida de audio. Todo lo útil o no que puede dar salida o entrada a las instrucciones de la computadora.

1. La memoria en una computadora es la que almacena datos a corto plazo, como la memoria Caché o la memoria RAM; podemos decir que esta es la memoria principal. Al apagar la computadora, se borrará todo lo existente en la memoria, pero es la que se encarga de los procesos actuales del CPU, mientras que el almacenamiento son los discos duros, pendrives o cualquier dispositivo que mantenga los datos a largo plazo.Esta vendría a ser la memoria secundaria.
2. La analogía del modelo inteligente es una forma de comparar un humano con un algoritmo y dar a entender el funcionamiento de una Inteligencia Artificial o el Machine Learning. Se usa para comprender mejor los fundamentos de lo antes mencionado diciendo que se comportan y actúan como un “modelo inteligente”. Sucede algo, lo procesas y hacés algo al respecto aprendiendo de eso.

Un modelo inteligente puede recibir un dato de entrada, el cual sería el suceso que vive el humano. A este dato que recibe lo procesa, analizando patrones y tomando una decisión. Qué hará a continuación con ese dato. De esto, queda el aprendizaje, la experiencia para tratar con casos similares ajustando sus parámetros para el futuro y acoplandola a sus conocimientos a medida que recibe información. Luego este modelo inteligente o este humano podrá desarrollar distintas tareas basándose en sus conocimientos y decisiones.

1. La arquitectura JVM consta de 3 elementos principales.

**Cargador de clases:** Es el encargado de cargar en la memoria principal las clases que se almacenaron en el bytecode del fichero .class generado.

**Runtime Area:** Es el que almacenará tus variables locales, los llamados a métodos, todos los objetos y sus variables de instancias y emitirá los errores que ocurran con cada uno.

**Motor de ejecución:** Es el que traduce el bytecode que generaste al lenguaje de máquina con un intérprete o compilador JIT según la necesidad a la vez que recolecta la basura no referenciada en el área del heap de la Runtime Area.