|  |  |
| --- | --- |
| **CICLO** | **: DESARROLLO DE APLICACIONES WEB A DISTANCIA** |
| **MÓDULO** | **: ENTORNOS DE DESARROLLO** |
| **ALUMNO** | **: ISMAEL SÁNCHEZ SÁNCHEZ** |
| **TAREA** | **: ED01** |

**8**

**1)** (1 p) Explica en qué aspectos está relacionada una aplicación informática con el hardware, y pon algún ejemplo de cómo optimizar el dicho software a la hora de programar.

Una aplicación informática se ejecuta sobre el hardware, el cuál necesita del software para resultar de utilidad. Las aplicaciones consumen distintos recursos del hardware durante su ejecución, siendo estos gestionados por el sistema operativo. Los programas pueden estar escritos en múltiples lenguajes distintos, pero el hardware sólo entiende binario. Por eso es necesaria una traducción de las instrucciones del programa.

**a.** Optimizar el uso del procesador.

Se consigue evitando pasos innecesarios en el código, estructurándolo correctamente, o escribiendo código que pueda ser ejecutado de forma paralela para aprovechar mejor los procesadores con múltiples núcleos.

**b.** Optimizar el uso de memoria.

Se puede conseguir implementando un “recogedor de basura”, un mecanismo que hace que el programa libere automáticamente las secciones de la memoria RAM que almacenan recursos que no van a volver a utilizarse a corto plazo.

**c.** Optimizar el uso de disco duro (almacenamiento y número de accesos)

Se puede conseguir eliminando texto, espacio, o caracteres innecesarios del código antes de la compilación, lo cual se puede conseguir de forma automática mediante el uso de algunos programas.

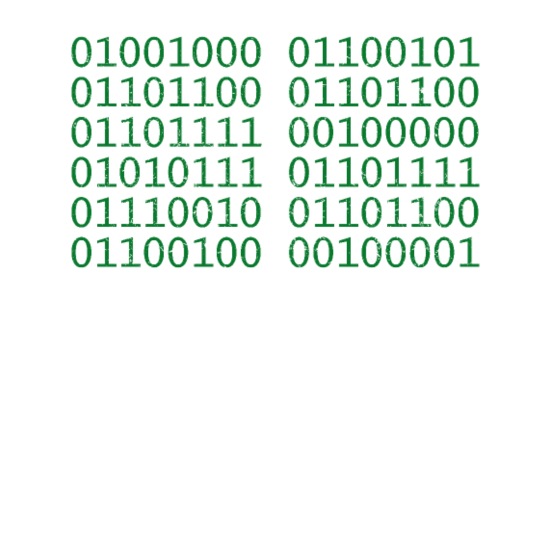
**2)** (0,5 p) Pon un ejemplo de software a medida y otro de software estándar.

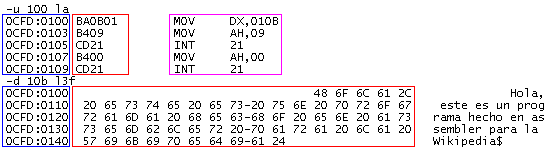
El software a medida son programas realizados para cumplir con las necesidades particulares de un cliente, y no suelen estar disponibles para un amplio público. Un ejemplo es Sabre, el software utilizado por la compañía American Airlines para gestionar y automatizar gestiones como reservas de vuelos, asignación de asientos, entre otras.

El software estándar está hecho para ponerse a disposición de un amplio público, y busca cubrir las necesidades generales que puedan tener los usuarios. Un ejemplo sería Microsoft Word, un editor de texto ampliamente disponible.

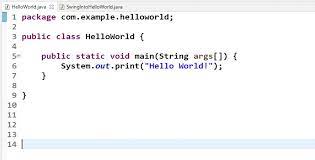
**3)** (0,5 p) Busca una imagen o pantallazo de un programa sencillo en lenguaje máquina, otro en ensamblador (indicando para qué procesador es) y otro de alto nivel. Así verás la diferencia entre ellos. (por ejemplo, el típico HOLA MUNDO en cada uno de ellos)

-Lenguaje máquina (código para escribir “Hello world” en pantalla):



-Lenguaje ensamblador para Intel 8088:

-“Hello world” en Java:



**4)** (1 p) Explica con tus palabras las diferentes maneras que hay de traducción de programas (compilado, interpretado y virtual), diciendo las principales características de cada una de ellas y sus ventajas y desventajas.

**Compilado**: En la traducción de programas de manera compilada, el código fuente escrito por el programador se pasa a través de un programa llamado compilador. Este compilador traduce el código fuente completo a un lenguaje de bajo nivel, a menudo en forma de código máquina específico de la arquitectura del procesador. Este código resultante es un archivo ejecutable que puede ser ejecutado directamente por la máquina.

Ventajas:

Eficiencia de Ejecución: El código compilado suele ejecutarse más eficientemente ya que ha sido traducido directamente a código máquina específico de la máquina objetivo.

Privacidad del Código Fuente: El código fuente no está disponible para el usuario final, lo que puede ser beneficioso para proteger la propiedad intelectual.

Desventajas:

Portabilidad Limitada: El código compilado es específico para una arquitectura de hardware y sistema operativo particular, lo que puede limitar su portabilidad a otras plataformas.

Proceso de Desarrollo más Lento: Dado que se requiere una fase de compilación antes de la ejecución, el desarrollo puede ser más lento.

**Interpretado**: En la traducción interpretada, el código fuente no se traduce completamente a código máquina antes de la ejecución. En cambio, el programa interpretador traduce y ejecuta el código fuente línea por línea en tiempo real.

Ventajas:

Portabilidad: El código fuente es generalmente independiente de la plataforma y puede ejecutarse en cualquier máquina que tenga el interpretador adecuado instalado.

Desarrollo Rápido: No se requiere un paso de compilación, lo que facilita un desarrollo más rápido y la corrección de errores de manera inmediata.

Desventajas:

Eficiencia de Ejecución: Por lo general, los programas interpretados tienden a ser más lentos en comparación con los programas compilados, ya que la interpretación agrega una capa adicional de procesamiento en tiempo real.

Privacidad del Código Fuente: El código fuente generalmente está disponible para el usuario final, lo que podría ser un problema en términos de propiedad intelectual.

**Virtual**: La traducción virtual implica el uso de una máquina virtual. En este enfoque, el código fuente se traduce a un código de byte intermedio que es ejecutado por una máquina virtual específica de la plataforma.

Ventajas:

Portabilidad: Similar a la interpretación, el código de byte intermedio es independiente de la plataforma y puede ejecutarse en cualquier máquina que tenga una máquina virtual adecuada.

Seguridad: Al ejecutarse en un entorno controlado por la máquina virtual, se puede lograr cierto nivel de seguridad.

Desventajas:

Eficiencia: Aunque puede ser más eficiente que la interpretación pura, aún puede ser más lento que la ejecución directa de código compilado.

Necesidad de la Máquina Virtual: Requiere la instalación de una máquina virtual en la máquina de destino.

**6)** (0,5 p) ¿Qué sabes de la traducción Just in Time? Explícala con tus palabras y pon ejemplo de lenguajes que utilicen esa traducción.

Se refiere a una técnica de programación en la que el código fuente se compila en unidades ejecutables según van siendo necesitadas por el programa. Se utiliza principalmente en Java y C#.

**7)** (1 p) Elige 3 lenguajes de programación de cada tipo compilados, interpretados y virtuales (también llamados mixtos o compilados-interpretados), es decir 9 en total. Indica de cada uno de ellos sus principales características y para qué suelen utilizarse.

**Compilados:**

C++

Características Principales: Es un lenguaje de programación de propósito general que combina características de programación orientada a objetos y de bajo nivel. Se compila directamente a código máquina.

Usos Comunes: Desarrollo de sistemas, videojuegos, software de alto rendimiento.

Rust

Características Principales: Diseñado para ser seguro, concurrente y práctico. Se enfoca en la prevención de errores y el rendimiento.

Usos Comunes: Programación de sistemas, desarrollo de software de alto rendimiento y seguro.

Java

Características Principales: Diseñado para ser portátil y orientado a objetos. Se compila a bytecode que se ejecuta en la máquina virtual Java (JVM).

Usos Comunes: Desarrollo de aplicaciones empresariales, desarrollo web (usando Java EE o Spring), aplicaciones móviles Android.

**Interpretados:**

Python

Características Principales: Lenguaje de alto nivel, fácil de aprender y leer. Interpretado y dinámicamente tipado.

Usos Comunes: Desarrollo web, ciencia de datos, inteligencia artificial, automatización de tareas.

JavaScript

Características Principales: Lenguaje de scripting para páginas web. Interpretado y ejecutado en el navegador.

Usos Comunes: Desarrollo web (front-end y back-end), creación de aplicaciones web interactivas.

Ruby

Características Principales: Lenguaje dinámico y orientado a objetos, con un enfoque en la simplicidad y la productividad.

Usos Comunes: Desarrollo web (con el framework Ruby on Rails), automatización de tareas, scripting.

**Virtuales (Compilados-Interpretados):**

C# (C Sharp)

Características Principales: Diseñado para la plataforma .NET, combina características de lenguajes compilados e interpretados. Se compila a bytecode que se ejecuta en la máquina virtual Common Language Runtime (CLR).

Usos Comunes: Desarrollo de aplicaciones Windows, desarrollo de juegos (con Unity), desarrollo web (usando ASP.NET).

Kotlin

Características Principales: Compatible con Java, se puede compilar a bytecode Java o ejecutarse en la máquina virtual de Kotlin (KVM).

Usos Comunes: Desarrollo de aplicaciones Android, desarrollo web (usando frameworks como Ktor), desarrollo de servidor.

Swift

Características Principales: Desarrollado por Apple, diseñado para ser seguro y eficiente. Puede ser compilado directamente a código máquina o ejecutarse en la máquina virtual Swift.

Usos Comunes: Desarrollo de aplicaciones iOS y macOS.

**8)** (0,5 p) ¿Cuáles son los lenguajes de programación más utilizados hoy en día y para qué se utilizan cada uno de ellos? (al menos 5)

Python: Se usa para el desarrollo de páginas web, inteligencia artificial, tareas de automatización, entre otros.

Javascript: Se usa para el desarrollo de aplicaciones web en entorno cliente, principalmente.

Java: Se usa para el desarrollo de aplicaciones Android, aplicaciones web, y aplicaciones para grandes empresas. Su portabilidad lo hace muy popular en este último caso.

C++: Se usa para software de sistemas, desarrollo de juegos, frameworks, y aplicaciones que requieren un bajo nivel de manipulación del hardware.

C#: Se usa en aplicaciones para Windows, desarrollo de juegos en Unity, y software empresarial.

**9)** (0,5 p) Indica la diferencia entre código fuente, código intermedio y código ejecutable.

El código fuente es aquel que escribe el programador directamente en un lenguaje de programación de su elección. Es altamente similar al lenguaje humano, no es ejecutable, ni entendible por la máquina. El código intermedio es un producto de los lenguajes intermedios. No es entendible ni por los humanos, ni por la computadora. Debe ser traducido a código máquina y enlazarse con las rutinas y bibliotecas necesarias para ser ejecutado. Una vez hecho esto, obtendremos el código ejecutable, el cual se cargará en la memoria RAM por el sistema operativo, ejecutándose así el programa, según las instrucciones del mismo.

**10)** (0,5 p) ¿Qué fases de análisis se realizan durante el proceso de compilación? Explica brevemente cada una de ellas.

Análisis léxico: Se extraen las cadenas de caracteres reconocidas como parte del vocabulario, generándose errores si no se pueden reconocer. Se genera un conjunto de tokens como salida.

Análisis sintáctico: Se toman los tokens obtenidos previamente, y se construye una representación intermedia de los mismos.

Análisis semántico: Se buscan problemas semánticos en la estructura obtenida en el análisis sintáctico.

Generación de código: Se transforma la representación intermedia en código máquina.

**11)** (1 p) ¿Cuáles son las fases de Desarrollo de software? Define cada una de ellas en 2 o 3 líneas como máximo.

Análisis: Se determinan los requisitos funcionales y no funcionales del software a desarrollar, y se diseña la arquitectura del mismo.

Diseño: División del sistema en partes, y establecimiento de las relaciones entre las mismas. Se toman decisiones como el lenguaje de programación a utilizar, o las entidades de las bases de datos.

Codificación: Se escribe el código fuente de los distintos módulos del programa.

Compilación: Se traduce el código fuente obtenido a código máquina

Pruebas: Utilizando un conjunto de prueba, se comprueba el funcionamiento del software a nivel unitario e integrado.

Explotación: Fase en que los usuarios finales conocen la aplicación y comienzan a utilizarla. Incluye instalación, puesta a punto y funcionamiento de la aplicación en el equipo final del cliente.

Mantenimiento: Contacto continuado con el cliente tras la explotación del software por parte del mismo, con el propósito de realizar los cambios, mejoras, o arreglos necesarios en el programa.

Documentación: Se debe mantener un registro de toda la información relevante al programa de forma simultánea al desarrollo del mismo durante todas sus etapas.

**12)** (1 p) Nombra los principales perfiles profesionales que participan en el desarrollo de software, indicando brevemente las funciones de cada uno de ellos.

Desarrollador de Software: Es el encargado de escribir el código fuente del software. Puede especializarse en áreas como desarrollo de front-end, back-end, o full-stack, y utiliza lenguajes de programación para implementar soluciones.

Analista de Sistemas: Analiza los requisitos del sistema, identifica problemas y propone soluciones. Colabora con los usuarios finales y el equipo de desarrollo para garantizar que el software cumpla con las necesidades y expectativas.

Arquitecto de Software: Diseña la estructura del software, define las tecnologías a utilizar y establece patrones de diseño. Su objetivo es garantizar la escalabilidad, rendimiento y mantenibilidad del sistema.

Ingeniero de Pruebas (Tester): Diseña y ejecuta pruebas para evaluar la calidad del software. Identifica errores, asegura la integridad de los datos y colabora en la mejora continua del proceso de desarrollo.

Propietario del producto: Representa al cliente y/o usuarios finales. Define las funcionalidades del producto, prioriza el trabajo del equipo y toma decisiones clave para el desarrollo del software.

Administrador de Bases de Datos: Diseña, implementa y mantiene bases de datos. Se encarga de la gestión eficiente de la información y garantiza la integridad y seguridad de los datos.

**13)** (0,5 p) Explica con tus palabras las diferencias que ves entre fase de análisis y de diseño. Valen ejemplos.

La principal diferencia entre las fases de análisis y diseño es que la fase de análisis se centra en qué debe poder realizar el programa, mientras que la fase de diseño se enfoca más en cómo cumplir con los requisitos especificados durante la fase previa. Otra forma de verlo, es que la fase de análisis determina primero los aspectos más generales del programa, y la fase de diseño profundiza y le da una forma más clara a los mismos. Por ejemplo, en la fase de análisis se puede determinar que un programa debe ser compatible con múltiples sistemas, y en la fase de diseño se decidirá usar un lenguaje de programación que facilite esta portabilidad.

**15)** Realiza una captura de pantalla, en la que se vea la realización de este trabajo y de fondo todo el escritorio de tu ordenador junto con la identificación en el aula virtual abierta. La captura tiene que ser personal y el trabajo que debe aparecer que sea en el procesador de texto que hayas elegido. Luego todo el documento, con la captura incluida, deberás exportarlo a PDF para subirlo a la plataforma.

