Congr. Int. en Ing. Electrónica. Mem. ELECTRO, Vol. 42 pp. 158-164, Oct 2020, Chihuahua, Chih. México
http://electro.itchihuahua.edu.mx/memorias_electro/MemoriaElectro2018.zip
ISSN 1405-2172

COMPARATIVA DE RESPALDO DE INFORMACIÓN BASADO EN LENGUAJE ENSAMBLADOR, CON RESPALDO EN LENGUAJES DE ALTO NIVEL

Serrano Ortega María Magdalena, Loreto Medina Claudia Selene, Gutiérrez Montoya Rosana, Romero Alvarado Karina Aidee
Tecnológico Nacional de México/Instituto Tecnológico de Tijuana
Departamento de Sistemas y Computación
Dirección: Calzada sn. Frac. Tomas Aquino Tijuana Baja California, México
Teléfono: 607 84 00
Magdalena.serrano@tectijuana.edu.mx, claudia.loreto@tectijuana.edu.mx,
rosana.gutierrez@tectijuana.edu.mx, kromero@tectijuana.edu.mx

RESUMEN.

El lenguaje ensamblador, es un lenguaje de programación de bajo nivel, su uso es potente, debido a que usa programación cercana al lenguaje máquina, implementando una representación simbólica de código maquina llamados mnemónicos, esta programación es muy usada en microprocesadores, microcontroladores y otros circuitos integrados programables, el ensamblador es necesario para programar procesador y constituye la representación más directa del código máquina específico para cada arquitectura.
El presente trabajo de investigación se realizó para demostrar la efectividad y la relevancia que tiene el lenguaje TASM(x86) hoy en día, para lograr esto se llevó a cabo el desarrollo de un programa capaz de realizar copias desde una unidad de almacenamiento externo hacia la unidad de almacenamiento principal del equipo. Contiene a su vez una descripción de los métodos y la planeación de desarrollo tomada para lograrlo, capturas con descripción del programa final, y una serie de comparaciones en distintos ámbitos entre el lenguaje TASM(x86) y otros lenguajes de nivel medio y alto.
Palabras Clave: Software, Mnemónicas, Metadatos, Modelo Jerárquico, Backup, Clúster, Partición, Encriptación, Datos, Multihilos, Formateo.

ABSTRACT.

The assembly language is a low level programing language, it use is powerful, because using programing close to the machine language, implementing a symbolic representation of machine code called mnemonics, this programing is widely usedin microprocessors, microcontrollers and others circuits integrated programmable, the assembler is necessary to program processor and constitutes the most direct representation of the specific machine code for each architecture.
The present research work was carried out to demonstrate the effectiveness and relevance of the TASM (x86) language today, to achieve this, the development of a program capable of making copies from an external storage unit to the equipment's main storage unit. It also contains a description of the methods and the development planning taken to achieve it, captures with a description of the final program, and a series of comparisons in different areas between the TASM (x86) language and other medium and high level languages.
Key Words: Software, Mnemonics, Metadata, Hierarchical Model, Backup, Cluster, Partition, Encryption, Data, Multithreading, Formatting.

1. INTRODUCCIÓN

Las capacidades de los lenguajes de programación de bajo nivel, como lo es ensamblador, son aún eficientes y usables para crear herramientas informáticas que sean capaces de resolver necesidades de cualquier usuario de un equipo de cómputo. En este caso en particular, un programa para ordenadores y/o computadoras portátiles basada en lenguaje ensamblador arquitectura x86 capaz de crear respaldos y copias de seguridad de un disco duro cuando el usuario lo desee, asegurando el correcto resguardo de la información; además de poder programar ejecuciones de la herramienta para que cada cierto tiempo, a criterio del usuario, pueda crear sus respaldos de sus discos duros automáticamente.

Esto surge bajo la necesidad de que cientos de miles de personas cuentan con acceso a dispositivos inteligentes como lo es la computadora, donde se realizan múltiples tareas y se maneja demasiada información. La necesidad de almacenar más y más información en nuestros equipos de cómputo ha provocado la creación de diferentes tipos de almacenamiento, como los habituales discos duros, y los más modernos, discos de estado sólido.

La manipulación de esta información conlleva al deseo de poder resguardarla de manera segura por ciertas razones, y una muy importante, por la pérdida parcial o total de la información. Es de relevante importancia para cualquier usuario de un ordenador respaldar la información de sus dispositivos de almacenamiento cada cierto tiempo, para que de esta manera pueda recuperarlos en caso de su pérdida.

Todo lo anteriormente mencionado bajo la justificación de intentar eliminar considerablemente los casos donde se pierde parcial o totalmente la información de un disco duro ya sea por formateos involuntarios, daños en el sistema, u otra razón; al tener un respaldo de dicha información para su recuperación. De igual manera, lograr que los usuarios usen una herramienta creada en lenguaje ensamblador, para que puedan observar la

importancia de conocer el mismo y en el mejor de los casos, la incitación del uso de este lenguaje que desfavorablemente está siendo olvidado por los nuevos programadores informáticos.

Software: Software es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador. [18]

Mnemónicas: Palabra que sustituye a un código de instrucción (Lenguaje de máquina). [19]

Metadatos: Son datos de definición que proporciona información acerca de la documentación o de otros datos gestionados dentro de una aplicación o entorno. [20]

Modelo Jerárquico: Conjunto de elementos de datos que se relacionan entre sí mediante relaciones jerárquicas. [21]

Backup: Copia de Respaldo o Seguridad. Acción de copiar archivos o datos de forma que estén disponibles en caso de que un fallo produzca la pérdida de los originales. [22]

Clúster: Unidad mínima de almacenamiento en un disco, donde el sistema operativo guarda la información. [23]

Partición: Es el nombre genérico que recibe cada división presente en una sola unidad física de almacenamiento de datos. [24]

Encriptación: Procedimiento de seguridad que consiste en la alteración, mediante algoritmos, de los datos que componen un archivo. [25]

Datos: Expresiones generales que describen características de las entidades sobre las que operan los algoritmos de una computadora. [26]

Multihilos: Programa que contiene dos o más partes que se pueden ejecutar de manera concurrente o simultánea. [27]

Formateo: Serie de operaciones realizadas con el fin de restablecer cualquier dispositivo que albergue datos, a su estado original. [28]

2. ESTUDIO ANALÍTICO DE HERRAMIENTA DE RESPALDO DE INFORMACIÓN BASADO EN LENGUAJE ENSAMBLADOR X86

2.1. Trabajos relacionados

El respaldo de datos es una práctica poco común entre empresas medianas y pequeñas implementación de esta práctica en conjunto de métodos de encriptación y seguridad pueden incrementar la seguridad de una empresa de una forma muy eficaz, aumentando la reputación de la empresa en los ojos de los clientes de esta. Un mal manejo de la información, por otro lado, puede llegar a llevar empresas a su fin. Es por esto que la implementación de un sistema de respaldos semi o completamente automático puede facilitar mucho el trabajo y mejorar la calidad de este. (Iron Mountain Inc. 2020)

3. “Uno de los elementos más importantes dentro de una organización son los datos, ya que al ser procesados generan información que ayuda a la toma de decisiones dentro de cualquier entidad. Debido a que esa información se encuentra almacenada en computadoras se corre el riesgo de perderla,

pero a la larga esta llega a ser muy útil, que se hace comúnmente de forma manual, es decir, sin ayuda de un software especializado en el traslado de datos, la esto puede ser ocasionado por fallos físicos tales como fallas en la corriente eléctrica, descompostura de los discos duros donde se encuentra almacenada dicha información, además existen problemas a nivel de software como son los virus, programas que realizan un comportamiento inusual provocando un fallo en los datos, también otro error muy común que suele suceder es la eliminación de datos por descuido de parte del usuario” (ZÚÑIGA ARCE, 2012, p. 3) .

Como vemos, son muchas las formas en las que podríamos perder nuestra información almacenada, que en muchos casos, son datos privados e incluso irrecuperables, como fotos familiares o de viajes. El uso de herramientas de análisis de archivos en conjunto de herramientas de respaldo, puede ayudar a determinar la mejor manera de manejar estos archivos, ya que existen diferentes tipos de archivos y documentos que deben ser resguardados (en uso, en reposo y en movimiento), el lograr identificar la diferencia entre estos puede ayudar a eficientizar en gran escala tanto los tiempos de respaldo como su espacio total. (B. H. Ali 2019)

Incluso muchos otros autores recomiendan por completo en continuo respaldo de seguridad de los datos, junto con esquemas de partición y cifrado de datos, siempre resaltando la importancia de los respaldos para proteger nuestra información. (Tian, 2020)

Entre las formas más óptimas de realizar copias de archivos se encuentra el aprovechamiento de los procesadores multi núcleos por medio de aplicaciones multihilos, esto debido a que

cada proceso de copiado ocupa uno de los núcleos por la duración de tarea necesaria; dependiendo claro del tamaño del archivo o complejidad del contenido. Es por esto que haciendo uso de la técnica multi-procedimiento se pueden copiar múltiples archivos a la vez, reduciendo exponencialmente el tiempo de ejecución necesario para concluir la misma tarea. (Zuojie Deng 2018)

Un problema muy común en el copiado de archivos desde una unidad de almacenamiento externa es el formato de cifrado, ya que este puede determinar si la información terminara siendo completamente inútil tras ser copiada o no, para esto se necesitan considerar dos opciones; 1) Que la unidad de almacenamiento no se encuentra encriptada o sea desencriptada antes de realizar el proceso de copiado. Y 2) Tener los medios y accesos necesarios para realizar la desencriptación de la información. (Uk Hur 2019).

3.1. Objetivos

- Desarrollo de algoritmos.
- Probar con los 3 algoritmos diferentes.
- Realizar un prototipo de software de copiado.
- Demostrar mediante tablas comparativas el mejor funcionamiento del software desarrollado en bajo nivel.

3.2. Metodología utilizada.

Dada la naturaleza del lenguaje ensamblador TASM, que requiere de menos recursos para poder hacer las operaciones**.** Por lo que a lo largo del desarrollo del programa nos enfocamos en replicar funciones básicas para realizar operaciones de memoria como, copiado, traslado, etc. en un disco duro.

La forma en la que comprobamos es a partir de varios sistemas de medición que indican los recursos ocupados, y el tiempo de operación, comparando entre un software especializado, las funciones de consola que ya están disponibles para este tipo de transacciones, y el programa que se desarrollara.

En base al desarrollo y análisis se presen a partir la información recabada una serie de gráficos y estadísticas al realizar la prueba; que se planea realizar utilizando varios discos duros de distintas capacidades y velocidades de lectura-escritura y diferentes grados de deterioro o uso (esto para verificar a qué medida o si es capaz de realizar estas operaciones en un medio dañado).

Partiendo de las estadísticas obtenidas se escribirán los resultados y la conclusión de la prueba, en la cual se dará un análisis extensivo de los resultados.

Con respecto al proceso que realiza nuestro software se describe en los siguientes pasos:

1. El primer paso es hacer que nuestro software propuesto realice una serie de operaciones y revisiones para identificar cuáles unidades de almacenamiento están disponibles.
2. Mostrar una lista de opciones de las unidades disponibles, para que el usuario elija.
3. Se pedirá al usuario que ingrese una dirección local para el almacenamiento de la copia. En caso de carpetas y subcarpetas; se planea empezar el proceso con la creación de la estructura de la unidad, ósea crear las carpetas necesarias para reflejar la misma estructura.
4. Para la última parte; el proceso de copiado en sí, se tendría que averiguar por medio del software el tamaño del archivo que se copiara, esto para evitar utilizar recursos innecesario.
5. Se realiza la copia en la dirección correspondiente.

3.3. Justificación.

El desarrollo de este proyecto se justifica ya que en base a la continua necesidad de mantener segura la información personal, empresarial y global, siempre se están buscando alternativas que garantizan la integridad de la información.

3.4. Funcionamiento del software.

Tabla 1. Tabla que ejemplifica el proceso del software propuesto

Actividad	Pasos a seguir
Identificación de unidad	1. El sistema identifica las unidades de almacenamiento del usuario y las enlista para el usuario. 2. Se solicita al usuario realizar una selección entre las unidades disponibles. 3. El usuario elige hacer una copia desde la unidad seleccionada.
Destino de copia	1. Al usuario se le pide escribir o elegir la ruta en la que se realizará la copia. 2. El usuario ingresa la ruta ejemplo: "C:\Users\Public\Pictures". 3. El programa confirma que la dirección exista, de no ser el caso crea las carpetas necesarias
Crear carpetas	1. En caso de que el usuario simplemente presiona "Enter" y deje la dirección vacía, el programa realizará la copia en la dirección de donde se encuentra el ejecutable. 2. El programa toma los nombres y crea las carpetas correspondientes solo en caso de que las carpetas no existan.
Copiado	1. Se asignan caracteres con la dirección interna, ósea de la estructura de las carpetas, y otros dos que se mantendrían siempre con

	el mismo valor, que serían. • Nombre de unidad de donde se hará la copia. • Dirección donde se realizará la copia. • Carpeta actual donde encuentra el archivo. 2. Se realiza copia de la carpeta fuente a la carpeta destino. 3. Se realiza una lectura de unidad y escritura de dirección de copiado. 4. La copia se realiza en segmentos

Fuente: Elaboracion Propia en base al desarrollo de nuestro proyecto

3.5. Propuesta de Investigación

Se desarrollaron un conjunto de algoritmos de diferente complejidad para probar el tiempo de compilación, la velocidad de ejecución y el tamaño del archivo en lenguaje ensamblador en competencia con lenguajes de alto nivel. Esto, con el fin de demostrar la premisa de que un algoritmo/programa desarrollado en lenguaje ensamblador en más eficiente en varios aspectos con respecto a sus precursores de alto nivel, las ejecuciones de los programas de ensamblador fueron en un emulador ya que se requiere de una arquitectura X86 y la mayoría de los equipos en la actualidad manejan procesadores mayores a 64 bits, por lo que, el tiempo de ejecución se vio afectado al no ser realizado propiamente en un equipo con arquitectura x86. De igual manera, los lenguajes de alto nivel tienen una ventaja significativa tomando en cuenta que sus compiladores, tienen métodos de optimización de código, por lo que podrían ser más veloces que sus predecesores.

1. El primer algoritmo desarrollado tiene una complejidad algorítmica de n. Concretamente, un aproximado de 60,000 operaciones para la finalización de su ejecución, se probo en 4 lenguajes diferentes: ensamblador, C#, Python y Java, ensamblador es mejor según resultados.

Lenguaje *Tiempo*	**Ensamblador x86**	**C#**	**Python**	**Java**
Tiempo de compilación	Instantáneo	0.76 segundos aprox.	0.72 segundos aprox.	0.50 segundos aprox.
Tiempo de ejecución	Instantáneo	Instantáneo	Instantáneo	Instantáneo
Tamaño del ejecutable	1 Kilobyte	5 Kilobytes	2 Kilobytes	3 Kilobytes

Fuente: Elaboración Propia en base al análisis del primer algoritmo probado en 4 lenguajes diferentes.

2. El segundo algoritmo desarrollado tiene una complejidad algorítmica de n^2. Concretamente, un aproximado de 3,600,000,000 operaciones para la finalización de su ejecución también se analizo en 4 lenguajes diferentes: ensamblador, C#, Python y Java, ensamblador es mejor según resultados.

Lenguaje Tiempo	**Ensamblador x86**	**C#**	**Python**	**Java**
Tiempo de compilación	Instantáneo	0.60 segundos aprox.	0.72 segundos aprox.	0.50 segundos aprox.
Tiempo de ejecución	3 segundos	12 segundos	24 minutos con 10 segundos	1 Segundo
Tamaño del ejecutable	1 Kilobyte	5 Kilobytes	2 Kilobytes	3 Kilobytes

3. El tercer algoritmo desarrollado tiene una complejidad algorítmica de n^3. Concretamente, un aproximado de 125,000,000,000 operaciones para la finalización de su ejecución se analizó en 4 lenguajes diferentes: ensamblador, C#, Python y Java, ensamblador es mejor según resultados.

Lenguaje Tiempo	**Ensamblador x86**	**C#**	**Python**	**Java**
Tiempo de compilación	Instantáneo	0.65 segundos aprox.	0.76 segundos aprox.	0.50 segundos aprox.
Tiempo de ejecución	3 minutos con 05 segundos	06 minutos con 40 segundos	1 hora con 15 minutos	01 minuto con 05 segundos
Tamaño del ejecutable	1 Kilobyte	5 Kilobytes	2 Kilobytes	3 Kilobytes

3.6. Propuesta del prototipo inicial

Para las instrucciones de copiado en el programa en lenguaje ensamblador (x86), el usuario debe elegir la ruta de origen, al usuario se le indican las unidades detectadas por medio de una lista, que las despliega con nombre y capacidad de almacenamiento para despues esperar a que el usuario inserte el nombre o ruta deseada como se muestra en la fig.1.

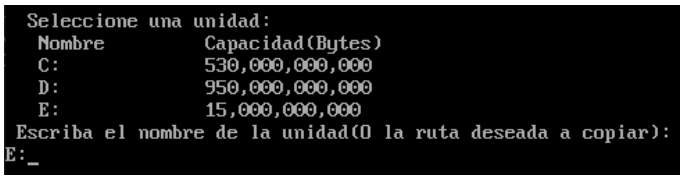

Fig. 1. Captura del usuario ingresando unidad origen.

El usuario debe ingresar la ruta destino y se esperara a que el usuario ingrese la informacion correspondiente (Fig. 2.).

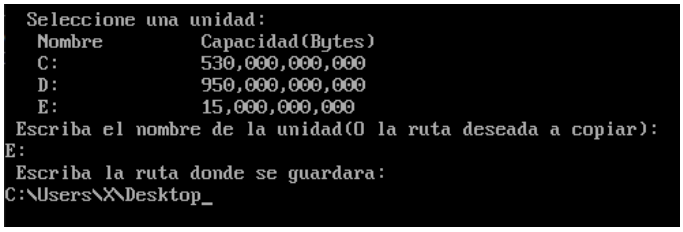

Fig. 2. Captura del usuario ingresando la ruta destino.

Se copiara la estructura desde origen a una carpeta con el nombre "CopiaUnidadX", a lo cual se crearan las carpetas necesarias para asimilar la estructura origen.

Fig. 3. Retroalimentación de creación de estructura.

Para que el usuario pueda tener retroalimentacion del punto del proceso en el cual se encuentra el programa, se desplegara el numero de archivo y byte que se encuentra siendo copiado por el programa en el momento, y despues de esto una barra de progreso con porcentage como se muestra en la figura 4.

Fig. 4. Retroalimentación de proceso.

Para que la aplicación mantenga una estructura simple y evitar que esta ocupe recursos innecesarios en tiempo de ejecución únicamente se actualizarán las partes necesarias a su momento, todo esto dentro de una sola pantalla de información.

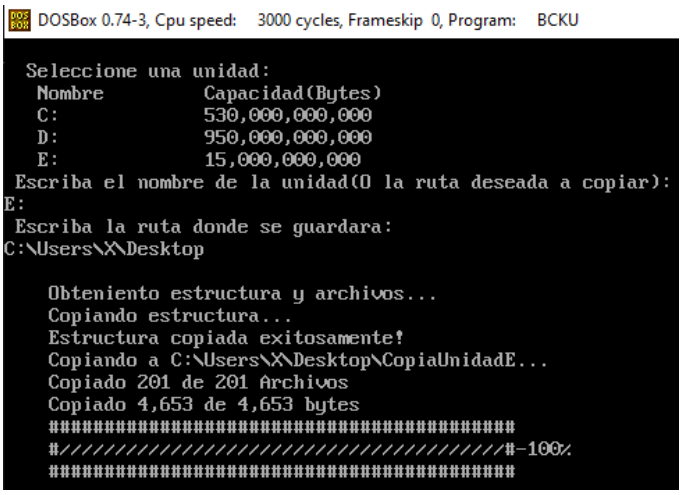

Fig. 5. Pantalla final.

3.7. Discusión de Resultados

En base al analisis comparativo de los 3 algoritmos, probados en 4 lenguajes de programacion diferente podemos indicar que de que las capacidades del lenguaje ensamblador son mejores con respecto a los lenguajes de alto nivel, con los que se comparo como C# o Python, Java por mencionar algunos, esto es debido aque ensamblador es un lenguaje de bajo nivel y no se esta considerando que las pruebas se realizaron bajo el ambiente de un emulador, lo que indica que supera por mucho nuestras espectativas.

En cuanto a la característica del tamaño del archivo, ensamblador es unánime ganador al usar muy poco espacio para la ejecución del mismo algoritmo.

El tiempo de compilación en ensamblador es instantáneo, al ya estar escrito en lenguaje de bajo nivel y no requerir de tiempo para traducirlo; en cambio, los compiladores en los otros lenguajes deben interpretar el código y traducirlo a un lenguaje de bajo nivel, para lo que necesitan tiempo, pero muy poco.

En el tiempo de ejecución se podrían apreciar dos cosas: primero, el lenguaje Java resultó ser el más veloz no solo en comparación con ensamblador, sino también con sus hermanos C# y Python, pero se puede entender esto al saber que, como se mencionó al inicio del apartado, los compiladores de los IDE's modernos tienen optimizadores de código, por lo que quizá puedan agrandar el tamaño del archivo, pero ser más veloces en su tiempo de ejecución. Segundo, los programas desarrollados en ensamblador, claro está que fueron ejecutados en un emulador al no disponer de un equipo con la arquitectura adecuada, por lo que su tiempo de ejecución se podría ver afectado al no estar en el mejor escenario posible.

Con estos resultados se podría llegar a la decisión que desarrollar herramientas para el respaldo de información u otras herramientas en un lenguaje de programación como ensamblador puede ser eficiente y viable al comprobar que sus capacidades pueden ser iguales o incluso mejores a la de los

lenguajes más moderno asi como optimizar procesos que en lenguajes de alto nivel requieren de tiempo y recurso.

3.8. Conclusión

El desarrollo de una herramienta de respaldo de información o cualquier herramienta de software basado en lenguaje ensamblador x86 resulta ser una alternativa que es interesante deexplorar debido a su posible implementación ya que se reducen mucho los tiempos y recursos de ejecucion.

Almomento de hacer uso de esta herramienta en bases de datos muy grandes o servidores masivos, podria ser interesante hacer una prueba ya que dentro de estos se busca un respaldo rapido y eficiente, sin perdida de informacion. Las copias de seguridad son requeridas en cualquier area, por lo que entre mas optimizado y mejores algoritmos desarrollados tengamos es mejor, además, de que la principal razón de que no se hagan es la falta de algún programa lo suficiente rápido como para superar las copias “manuales”.

También obtuvimos resultados bastante aceptables en la velocidad de trabajo del lenguaje ensamblador, tomando en cuenta que se realizaron las pruebas en un emulador.

Como trabajos a futuro, esta la posibilidad de programar la herramienta de respaldo mencionada en este estudio de forma inmersa, es decir mezclando ensamblador con alguno de los lenguajes de alto nivel mostrado.

3.9. Referencias

[1]. Definición de Ensamblador. (2010). Recuperado 14 de mayo de 2020, de sistemas.com/ensamblador.php

[2]. C. E. (2017, mayo). Lenguaje ensamblador - Algorítmica y Programación. Recuperado 14 de mayo de 2020, de sites.google.com/site/portafoliocarlosmacallums/unidad-i/lenguajeensamblador

[3]. H. P. (2019). DEFINICIÓN DE ALMACENAMIENTO DE DATOS. Recuperado 14 de mayo de 2020, de hpe.com/mx/es/what-is/data-storage.html

[4]. Samsung. (2018). SSD 860 QVO 1TB. Recuperado 14 de mayo de 2020, de samsung.com/es/memory-storage/860-qvo-sata-3-2-5-ssd/MZ-76Q1T0BW

[5]. Red Hat. (2018). Data storage: Dispositivos de almacenamiento de datos. Recuperado 14 de mayo de 2020, de www.redhat.com/es/topics/data-storage

[6]. Artman, J. (2012). Discos duros de 5400 RPM vs. 7200 RPM. Recuperado 14 de mayo de 2020, de techlandia.com/discos-duros-5400-rpm-vs-7200-rpm-info_132496

[7]. Mártil, I. (2017, mayo). ¿Cómo se guarda la información en un ordenador? una tarea asombrosa en la era digital. Recuperado 14 de mayo de 2020, de blogs.publico.es/ignacio-martil/2017/05/12/como-se-guarda-la-informacion-en-un-ordenador-una-tarea-asombrosa-en-la-era-digital/

[8]. Video, M. (2019, julio). Dispositivos de almacenamiento de información. Recuperado 14 de mayo de 2020, de marcass.com.mx/dispositivos-de-almacenamiento-de-informacion

[9]. Tecnologiafacil. (2017, marzo). Respaldo de información. Recuperado 14 de mayo de 2020, de tecnologia-facil.com/como-hacer/respaldo-de-informacion/#Que_es_respaldo_de_informacion_Definicion

[10]. Raffino, M. E. (2020, enero). Backup: Concepto, Usos y Cómo hacer backups. Recuperado 14 de mayo de 2020, de https://concepto.de/backup/

[11]. ZÚÑIGA ARCE, J. A. (2012, octubre). AUTOMATIZACIÓN DE LOS PROCESOS DE RESPALDO DE INFORMACIÓN DEL TALLER DE DESARROLLO DE SOFTWARE DEL DEPARTAMENTO ACADÉMICO DE SISTEMAS COMPUTACIONALES. TÍTULO, 1-72.

[12]. Tian, J. (2020, mayo). Low-cost data partitioning and encrypted backup scheme for defending against co-resident attacks. Recuperado de jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-020-00110-1

[13]. Zuojie Deng,Shuhong Chen,Xiaolan Tan,Dan SongFan Wu (2018 Diciembre) An Efficient Provable Multi-copy Data Possession Scheme with Data Dynamics ,Recuperado de springer.com/chapter/10.1007/978-3-030-05345-1_34

[14]. Uk Hur, Myungseo Park, Giyoon Kim, Younjai Park, Insoo Lee, Jongsung Kim (Diciembre 2019) Data acquisition methods using backup data decryption of Sony smartphones, Recuperado de sciencedirect.com/science/article/abs/pii/S1742287619301495

[15]. Iron Mountain Incorporated (2020, enero) MEJORES PRÁCTICAS ,Recuperado de ironmountain.com.mx/resources/data-sheets-and-brochures/f/five-best-practices-for-protecting-backup-data

[16]. Basheer Husham Ali, Ahmed Adeeb Jalal, Wasseem N. Ibrahem Al-Obaydy (2019, marzo) Data loss prevention by using MRSH-v2 algorithm Recuperado de researchgate.net/profile/Ahmed_Jalal7/publication/339375903_Data_loss_prevention_by_using_MRSH-v2_algorithm/links/5e4fa47a92851c7f7f492fc1/Data-loss-prevention-by-using-MRSH-v2-algorithm.pdf

[17]. LWP, Mnemónico, 2020, de La Web del Programador Sitio Web : https://www.lawebdelprogramador.com/diccionario/Mnemonico/

[18]. LWP, Mnemónico, 2020, de La Web del Programador Sitio Web: https://www.lawebdelprogramador.com/diccionario/Mnemonico/

[19]. LWP, Metadatos, 2020, de La Web del Programador Sitio Web: https://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=Metadatos

[20]. EcuRed (2019), Datos jerárquicos, 2020, de EcuRed Sitio Web: https://www.ecured.cu/Datos_jer%C3%A1rquicos

[21]. LWP, Backup, 2020, de La Web del Programador Sitio Web: https://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=Backup

[22]. LWP, Clúster, 2020, de La Web del Programador Sitio Web: https://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=Cl%C3%BAster

[23]. EcuRed (2018), Partición de disco, 2020, de EcuRed Sitio Web: https://www.ecured.cu/Partici%C3%B3n_de_disco#:~:text=Partici%C3%B3n%20de%20disco.,f%C3%ADsica%20de%20almacenamiento%20de%20datos.

[24]. Significados, Encriptación, 2020, de Significados Sitio Web: https://www.significados.com/encriptacion/#:~:text=La%20encriptaci%C3%B3n%20es%20un%20procedimiento,que%20un%20tercero%20los%20intercepte.

[25]. DefiniciónDE, Datos, 2020, de DefiniciónDE Sitio Web: https://definicion.de/datos/#:~:text=Para%20la%20inform%C3%A1tica%2C%20los%20datos,ser%20tratadas%20por%20una%20computadora.

[26]. DevMagazine (2018), Conoce todo acerca la programación en Multihilo, de DevMagazine Sitio Web: https://devmagazine.co/conoce-todo-acerca-la-programacion-en-multihilo/2008/#:~:text=Un%20programa%20multihilo%20es%20aquel,especializada%20para%20ejecutar%20una%20multitarea.

[27]. ConceptoDefinición (2019), Formatear, de ConceptoDefinición Sitio Web: https://classroom.google.com/u/2/c/MTUxNTMwNTczOTRa

[28]. LWP, Software, 2020, de La Web del Programador Sitio Web: https://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=Software

Se deberán listar y numerar todas las referencias al final del artículo empleando para ello la norma de citación de IEEE con el estilo *Referencia* de la siguiente manera:

[1] N.R. Vela, Titulo, Ciudad, Editorial, año, páginas

[2] G.M. Cotty, Título, Revista, Vol., mes año, páginas.

[3] H. Kwak, "Título", organización [en línea], disponible: liga de Internet, sitio visitado fecha.

[4] B. Pérez, "Título", Congreso o Conferencia, ciudad, año, páginas.