UNIDAD TEMÁTICA 3
El software de base. Sistemas operativos y lenguajes.

Software de base: concepto.
Sistema operativo: concepto y funciones. Sistemas operativos para PC.
Programas utilitarios: concepto, clasificación.
Lenguajes de programación: niveles, paradigmas, orientaciones, traductores.

3.1) SOFTWARE

Todo computador trabaja sobre la base de un programa (conjunto de instrucciones ordenadas en una secuencia predeterminada, siendo cada instrucción una orden que se imparte al computador indicándole lo que debe hacer y usar para llevar a cabo una tarea).

Se denomina software al conjunto de programas y se lo clasifica en:

a) Software de Base.
b) Software de Aplicación o Aplicativo.

Software de Base: Es toda aquella parte lógica realizada generalmente por cada fabricante de computadores o por casas especializadas en el desarrollo de software de base.

Este conjunto de programas tiene por función coordinar las diversas partes del sistema computacional para hacerlo funcionar rápida y eficazmente, actuando como mediadores entre los programas de aplicaciones y el hardware del sistema, interpretando los requerimientos de cada programa que ingresa al sistema, poniendo a su disposición cualquiera de los recursos que necesite (ya sean de hardware, software o datos) para producir los resultados deseados.

Debido a la estrecha relación que existe entre las características de un computador y su software de base, no se puede concebir el uno sin el otro, a tal punto que pareciera que el software es parte integrante del hardware.

El software de base se suele clasificar en:

a) Sistema Operativo
b) Utilitarios
c) Traductores de lenguajes

3.2.1) Sistema Operativo: Concepto y Funciones

Concepto: Es un conjunto de programas concebidos para efectuar la administración de los recursos del computador.

Del conjunto de instrucciones que maneja el computador, algunas residen permanentemente en la memoria central (luego de la ejecución del programa de carga inicial -IPL-) durante todo el procesamiento, mientras que otras residen solo cuando se las necesita, encontrándose almacenadas en periféricos cuando no están en la memoria central. A las primeras se las conoce como residentes, supervisor, monitor y ejecutivo. A las segundas como transcientes.

Funciones:

El Sistema Operativo tiene dos funciones básicas, globalmente consideradas:

1) Proveer servicios para la ejecución de programas de aplicación y para el desarrollo de los mismos, es decir administrar los recursos en proceso, obtener automáticamente la rutina apropiada y mantener el computador sin necesidad de operación manual.

2) Actuar como entorno de la aplicación en la cual el programa es ejecutado, planeando los recursos y trabajos, puesto que ayuda a decir no solo qué recursos utilizar (asignación), sino también cuándo utilizarlos (planificación). Debido a que los dispositivos de Entrada-Salida trabajan mucho más lentamente que la C.P.U., pueden realizarse millones de instrucciones de cálculo para varios programas, mientras que los resultados se imprimen o muestran por pantalla.

Utilizando varias técnicas el Sistema Operativo combina los diversos trabajos que deben realizarse de modo que los dispositivos del sistema se empleen lo más eficientemente posible.

Las instrucciones que conforman un Sistema Operativo, materializan entre otras, las siguientes tareas en la ejecución de una aplicación:

Las instrucciones que conforman un Sistema Operativo, realizan entre otras, las siguientes tareas en la ejecución de una aplicación:

- Carga de programas.
- Gestión del tiempo de procesamiento.
- Gestión de la memoria principal.
- Gestión de la memoria secundaria (ficheros y directorios).
- Gestión del subsistema de e/s (drivers).
- Seguridad y protección del sistema.
- Interfaz de llamadas al sistema.
- Interfaz de usuario y utilidades del sistema.
- Tareas de comunicación de datos (teleprocesamiento).

Generalmente en algunos equipos el Sistema Operativo cumple funciones de monitoreo: lleva registro de las actividades del computador mientras se realiza el procesamiento. El Sistema Operativo detiene los programas que contienen errores o exceden, ya sea su tiempo máximo de ejecución o sus asignaciones de almacenamiento. Mediante el envío de mensajes informa las anormalidades en los dispositivos de Entrada-Salida o en otra parte del sistema. Son también parte del Sistema Operativo la contabilización o registro de hora de ingreso y egreso, y el tiempo de duración de los programas, lo que hace posible elaborar facturas por concepto de utilización del sistema por parte de los usuarios.

Posee además mecanismos de seguridad para proteger contra el acceso no autorizado a través de la verificación de identificación ("claves" o “passwords”).

3.2.2) Sistemas Operativos para PC.

La primera IBM-PC aparece con un sistema operativo desarrollado por Microsoft, denominado DOS 1.0. El mismo administraba la PC con tan solo disketteras. Aparece luego la versión 2 que controlaba también discos rígidos. La 3 agrega posibilidades de compartir dispositivos (en una red). Luego vinieron las versiones 4, 5 y 6, que fueron incorporando más utilidades (compresión de discos, resguardo, verificadores, ayudas, interfase gráfica elemental, etc..). También han sido desarrollados otros sistemas por otras compañías, como por ejemplo Digital Research que desarrolló el DR DOS cuya primera versión fue la 5.0, le siguieron la 6 y la 7, también agregando más utilidades y mejor integración a ambientes de red.

Asimismo y en forma paralela, Microsoft comienza el desarrollo de un entorno operativo que permitía una interfase gráfica mas sencilla e intuitiva para el usuario, tomando como ideas los desarrollos realizados por un área de la empresa Xerox, los que también han sido llevados a otras arquitecturas de equipos como las Apple Lisa y

Macintosh.

Este entorno se ejecutaba por encima del DOS, y no tuvo una significativa aceptación en sus versiones 1 y 2. No obstante las mejoras introducidas a la versión 3 y la mayor potencia que se tenía en el hardware, posibilitaron su gradual utilización. En realidad constituyó todo un suceso, que se afirmó con las siguientes versiones (3.1, 3.11 y 3.11 para Grupos de Trabajo). Paralelamente Microsoft estaba trabajando con IBM en el desarrollo de un nuevo sistema operativo gráfico, denominado OS/2. Por distintas motivaciones, IBM se desvincula de Microsoft en este proyecto y lo continúa sola, ofreciendo luego comercialmente a este producto. Por su parte Microsoft desarrolla una nueva versión de Windows que a diferencia de las anteriores es un sistema operativo y no solo un entorno. Guarda compatibilidad con las anteriores pero tiene significativas mejoras, tanto en lo estético como en lo funcional. La denominó Windows 95. Lanza luego otra versión destinada a un segmento de equipos de mayores requerimientos y/o administración de recursos en red de área local, que denominó Windows NT. La versión menor de Windows, la 95, tuvo una actualización denominada Windows 98, que continuó con la orientación de la 95, mejorando performance, y agregando utilidades. Por su parte la mayor, NT, también ha ido evolucionando, apareciendo las versiones 3,5, y 4. A partir de estos últimos, Microsoft ofrece una familia de productos basada en su Sistema Operativo (dependiendo de los requerimientos del usuario NT Workstation para equipos autónomo con un único usuario con altos requerimientos o NT Server para servidores de red, administrando redes de área local, preferentemente en modo dedicado).
En el segmento menor, Microsoft desarrolla para el año 2000 una versión que denominó Millenium, y para el segmento mayor actualiza el NT con una versión que denominó 2000.

3.3.) Programas utilitarios: concepto, clasificación.

Llamamos ***utilitarios*** a aquellos programas entregados por el fabricante, comprados a terceros o desarrollados en la propia instalación, de uso general en todo equipo, escritos con el objeto de realizar tareas repetitivas de procesamiento de datos.

Estas tareas se realizan con tanta frecuencia en el curso del procesamiento, que sería extremadamente ineficiente el que cada usuario tuviera que codificarlas en forma de programas una y otra vez.

Desde el punto de vista de las funciones que cumplen, los podemos agrupar en:

- Utilitarios de apoyo a los sistemas de aplicación: Estos programas se integran al sistema de aplicación, es decir, que su función formar parte de la secuencia de procesamiento necesaria para operar el sistema de aplicación; por ejemplo: generador de copias de archivo, generador de listados, clasificador e intercalador de archivos, etc.

- Utilitarios de Servicios: Por un lado se incluyen en este grupo un conjunto de utilitarios que ayudarán a manejar ciertos recursos del computador, y por otro a los utilitarios para el manejo de programas y sus bibliotecas; por ejemplo: listador del directorio de un disco, inicializador de discos, diskette, cinta, cassette, el que elimina o renombra archivos, el reorganizador de espacios en discos, los compiladores y compaginadores, etc.

3.4. Lenguajes de programación: niveles, paradigmas, orientaciones, traductores

Un lenguaje es el conjunto finito de símbolos básicos permitidos, combinados de acuerdo con ciertas reglas del lenguaje a las que se denominan reglas de sintáctica.

En los primeros días de la computadora, a fines de la década de 1940, cada programa (o sea la serie de instrucciones que indica a la computadora el trabajo que se va a hacer) tenía que estar escrito en lenguaje de máquina. El único que una computadora puede entender directamente y que consta de combinaciones de ceros y unos.

Todos los usuarios tenían que escribir programas compuestos de largas cadenas de ceros y unos para especificar numéricamente la dirección de los datos y los códigos de operaciones que se debían ejecutar en la máquina.

Varios años mas tarde, se desarrollaron programas llamados traductores, los cuales aceptaban como entrada cierto lenguaje simbólico o mnemotécnico para luego convertirlo automáticamente en lenguaje de máquina.

Estos traductores se conocen como ensambladores, que, aunque ahorraban al usuario mucho trabajo, no eran lo suficientemente atractivos para ellos puesto que resultaba molesto tener que especificar, aunque simbólicamente, direcciones y códigos de operaciones.

Para resolver problemas, uno tenía que programar todavía en un lenguaje parecido al de máquina.

Estos lenguajes reciben el nombre de lenguajes de bajo nivel, debido a que, como dijimos anteriormente, los programadores debían escribir instrucciones con el mas fino nivel de detalle dado que la traducción que se realiza es uno-a-uno (cada línea de código corresponde a una sola acción del sistema computacional).

Los siguientes lenguajes que aparecieron fueron los lenguajes de alto nivel en los que se introduce el concepto de macroinstruccción (la traducción es una instrucción de alto nivel a muchas de bajo nivel, una-a-muchas).

Dentro de esta categoría se encuentran lenguajes tales como BASIC, COBOL, FORTRAN, PASCAL, PL/1, APL, C, etc.

Los lenguajes de alto nivel difieren de sus antecesores de bajo nivel en que requieren menos detalle de codificación. Los traductores que convierten el programa escrito en lenguaje de alto nivel al lenguaje de máquina proporcionan el detalle.

Como resultado los programas escritos en lenguaje de alto nivel son menos extensos y mas fáciles de escribir que aquellos escritos en lenguaje de bajo nivel.

Los lenguajes de muy alto nivel, que aparecieron por primera vez en la década de 1960, se crearon para cubrir necesidades especializadas del usuario y son relativamente fáciles de aprender y de utilizar por lo que se los denominan “amigables” para el usuario. Con los lenguajes de muy alto nivel solo se necesita prescribir lo que la computadora hará en vez de como hacerlo.

Existen muchos lenguajes de muy alto nivel en el mercado y por lo general hay más de uno por cada tarea de aplicaciones:

- Generadores de informes (DMS, RPG).
- Generadores de programas (se los conoce como 4to.nivel).
- Software para procesamiento de palabras.
- Hojas o planillas electrónicas.
- Paquetes de graficación.

etc..

La tendencia es acortar la brecha de comunicación entre hombre y máquina permitiendo que los no especialistas usen la computadora en un amplio número de disciplinas y prueben sus beneficios.

Con la venida de la nueva tecnología y de la nueva generación de computadoras, los lenguajes y sistemas en línea han sido y están siendo desarrollados para interactuar más como le gusta al hombre: rápidamente y de un modo conversacional.

LENGUAJES ORIENTADOS AL PROBLEMA Y AL PROCEDIMIENTO.

Los lenguajes de bajo y alto nivel se conocen como lenguajes de procedimientos, debido a que requieren que las personas escriban procedimientos detallados que indiquen a la computadora como realizar tareas individuales.

Los lenguajes de muy alto nivel, en contraste, reciben el nombre de lenguajes orientados al problema puesto que cada uno fue creado para resolver un problema en especial.

Además, en un amplio rango de aplicaciones, es fácil distinguir si la misma tiene características 'administrativo-contables' o 'científico-técnicas'.

Las primeras se caracterizan por requerir el manejo de un número elevado de datos, normalmente organizados en archivos, y realizar pocas operaciones sencillas con ellos. Por el contrario, las aplicaciones científico-técnicas utilizan comparativamente menor número de datos pero realizan un mayor y más complejo cálculo con ellos.

Muchos lenguajes de alto nivel o evolucionados tuvieron en cuenta estos aspectos y por lo tanto se encontraban orientados para cumplir mas eficientemente alguno de los dos tipos de procesamiento tipificados anteriormente. La evolución que luego han sufrido estos lenguajes ha hecho que paulatinamente se tornen más aptos para cualquier tipo de procesos, aunque mantienen su mejor predisposición para el cual fueron diseñados.

Así por ejemplo el COBOL surge como un lenguaje para resolver los problemas del área administrativa y el FORTRAN lo hace para el área científica.

BASIC: Características del lenguaje. Estructura del programa. Definición de datos. Enunciados.

El BASIC, cuyo nombre proviene de las siglas **B**eginner's **A**ll-purpose **S**ymbolyc **I**nstruction **C**ode (código de instrucción simbólica de uso general para principiantes), es un lenguaje fácil de aprender y que, al paso de los años, se ha convertido en uno de los lenguajes de programación más populares y de más fácil adquisición en los proveedores especializados.

Debido a que las necesidades de almacenamiento de su traductor de lenguaje son pequeñas, trabaja con eficiencia en casi todas las computadoras personales.

Existen muchas versiones del lenguaje BASIC, desde las simplificadas, que se utilizan en computadoras de bolsillo, hasta las poderosas versiones para computadoras a gran escala que compiten con el poder de procesamiento del COBOL.

En BASIC cada instrucción se suele identificar con un número de línea: por ejemplo 10, 20, 30, etc.

La computadora siempre ejecutará las instrucciones en la secuencia especificada por los números de líneas a menos que se ordene lo contrario mediante las instrucciones de ruptura de secuencia (IF, GOTO, GOSUB, etc.).

Cada instrucción comienza con una palabra clave, la cual indica a la computadora que tipo de operación debe realizar: por ejemplo REM, READ, LET, PRINT, DATA, etc. Estas palabras claves pueden considerarse el vocabulario del sistema computacional cuando se escriben programas en lenguaje BASIC.

Uno debe siempre apegarse estrictamente a este vocabulario. Si, por ejemplo, se sustituye DATA por DATE, la computadora no sabrá que es lo que uno quiere que haga.

A pesar de sus muchas ventajas, una debilidad importante que presentan muchas versiones de este lenguaje es que no están diseñadas para facilitar la programación estructurada.

Un programa largo y no estructurado escrito en BASIC puede resultar difícil de seguir. Asimismo, ya que existen tantas versiones del lenguaje BASIC, un programa desarrollado en una computadora puede requerir modificaciones sustanciales para ejecutarse en otra máquina o en otra versión de traductor o intérprete del lenguaje.

COBOL. Características del lenguaje. Estructura del programa. Divisiones.

El COBOL, cuyo nombre proviene de las siglas **CO**mmon **B**usiness **O**riented **L**anguaje (lenguaje común orientado a los negocios), fue introducido por primera vez en los inicios de la década de 1960.

Casi todas las características principales del COBOL se relacionan con su orientación al procesamiento de datos de negocios, incluso la independencia de la máquina, la autodocumentación, y la orientación a la entrada y salida.

Independencia de la máquina: es un aspecto importante ya que los programas para el procesamiento de datos de negocios generalmente tienen que durar mucho tiempo (10 o incluso 20 años).

Durante este período, una organización puede comprar nuevo hardware o cambiar completamente de un sistema computacional a otro. De este modo, los programas escritos para un sistema deben poder ejecutarse en otros con pequeñas modificaciones.

Autodocumentación: debido a que los programas de procesamiento de datos de negocios deben durar un largo tiempo, necesitan mantenimiento continuo. Por ello, es extremadamente importante que la lógica del programa sea fácil de seguir por otros programadores o aún por el mismo que lo codificó después de transcurrido un período de tiempo.

El lenguaje COBOL se presta para un buen diseño de programas en tres formas: legibilidad, modularidad, y uso adecuado de las tres estructuras básicas de control de la diagramación estructurada: (Secuencia, bifurcación e iteración).

El lenguaje COBOL también utiliza verbos del idioma inglés (como SUBTRACT, MOVE, ADD, etc.) y conectivos (como FROM, GIVING, etc).

Orientación a la entrada y salida: el procesamiento de datos de negocios, en contraste a las aplicaciones científicas y de ingeniería, implica la manipulación de grandes archivos con muchos registros.

Así, gran parte del trabajo en aplicaciones del procesamiento de datos de negocios se relaciona con la lectura y escritura de registros, y el lenguaje COBOL se ha diseñado para ser particularmente efectivo en esta tarea.

Contiene estipulaciones para definir de manera explícita y fácil el formato de los registros de entrada y salida. Por ejemplo, es un proceso muy sencillo el de editar cantidades monetarias , la salida con signos, puntos decimales, comas, y también redondear las cantidades.

Estructura del programa: todo programa escrito en lenguaje COBOL se agrupa en cuatro divisiones:

- División de identificación: en la que se identifica el nombre del programa, el autor, fecha de escritura y otros detalles. Esta división existe principalmente con fines de documentación.
- División de ambiente: en la que los nombres de archivos creados por el programador se vinculan a un equipo específico de entrada/salida. Aquí, por ejemplo, el programador especificaría que un archivo de entrada en particular, digamos ARCHIVO-DISCO, se localiza en disco y que un archivo de salida en particular, como ARCHIVO-IMPRESION, se dirigirá a la impresora.
- División de datos: en la que el programador nombra y define todas las variables del programa e indica su relación mutua.
- División de procedimientos: en la cual se especifican los procedimientos reales que la computadora debe seguir para crear la salida deseada.

Las tres primeras divisiones aseguran que todas las especificaciones importantes se establezcan en forma explícita en el programa.

Desventajas: Los programas escritos en lenguaje COBOL tienden a ser extensos y además se necesita un traductor de lenguaje grande y complejo para convertir los programas en el lenguaje de máquina, lo cual hace al COBOL difícil de implantar en computadoras pequeñas.

Por lo general no resulta adecuado para aplicaciones científicas y de ingeniería, las cuales utilizan demasiadas fórmulas complicadas.

Otros lenguajes de programación. Características generales.

FORTRAN cuyo nombre proviene de **FOR**mula **TRAN**slator (traductor de fórmulas), data del año 1954 y es el lenguaje comercial de alto nivel superviviente más antiguo.

Fue diseñado por científicos y está orientado hacia la resolución de problemas científicos y de ingeniería. La principal característica del FORTRAN es su capacidad para expresar con facilidad fórmulas complicadas. Aunque el BASIC es competitivo en esta tarea, el FORTRAN es generalmente superior para muchas aplicaciones debido a que hace posible una ejecución más rápida del programa y una mayor precisión, aunque hay versiones recientes del BASIC que alcanzan y aún superan sus prestaciones.

El FORTRAN por lo general utiliza un compilador como traductor del lenguaje.

Los compiladores ejecutan los programas más rápido que los intérpretes, que se utilizan en muchas de las versiones BASIC.

La lógica de los programas escritos en FORTRAN es más difícil de seguir que la lógica de algunos otros lenguajes, y es claramente inferior al COBOL para aplicaciones de procesamiento de datos de negocios.

PASCAL es un lenguaje relativamente nuevo, creado hacia 1970 para cubrir la necesidad de contar con una herramienta para la enseñanza de la programación estructurada.

Los compiladores del lenguaje PASCAL son extremadamente pequeños, lo que facilita la implementación de este lenguaje en la mayoría de las computadoras personales.

No obstante este lenguaje no resulta tan adecuado como el COBOL para las aplicaciones de procesamiento de datos de negocios y para complicadas operaciones aritméticas es superado por el FORTRAN y el BASIC.

CONCEPTO DE PROGRAMA. PROGRAMA FUENTE Y PROGRAMA OBJETO. COMPILADORES: CONCEPTO Y FUNCIONES.

Como ya se mencionó, las computadoras pueden ejecutar programas solo después de que estos han sido traducidos al lenguaje de máquina.

Hay dos motivos por los cuales las personas generalmente no escriben programas en este lenguaje:

Primero, las instrucciones del lenguaje de máquina constan de cadenas de apariencia compleja de ceros y unos. Por ejemplo:

0100111101010101010101010000111

Segundo, las instrucciones en el lenguaje de máquina deben ser escritas en el nivel de exposición más detallado. Por ejemplo, la computadora no puede sumar directamente A y B, colocando el resultado en C, con una sola instrucción como

C = A + B

Aún una simple tarea como ésta requiere tres o más instrucciones en lenguaje de máquina, como:

1. Cargar el valor representado por A de la memoria principal en un registro.
2. Sumar el valor representado por B de la memoria principal en el mismo registro.
3. Colocar la suma obtenida en otra zona de almacenamiento.

Estas instrucciones detalladas, a veces se denominan microinstrucciones, ya que no pueden subdividirse en comandos más pequeños. Una instrucción como C = A + B, por otro lado, es un ejemplo de macroinstrucción.

Las macroinstrucciones deben ser divididas en microinstrucciones por el sistema computacional antes de ser procesadas.

Todos los lenguajes de alto nivel (como BASIC, FORTRAN y COBOL) utilizan este tipo de instrucciones para ahorrar al operador la tediosa tarea de explicar en detalle a la computadora como hacer el trabajo.

Un traductor de lenguaje es simplemente un programa de sistemas que convierte un programa con macroinstrucciones en uno con microinstrucciones en base binaria.

Los tipos de traductores de lenguajes son: ensamblador, compiladores e intérpretes.

ENSAMBLADORES:

El ensamblador, se utiliza exclusivamente con los lenguajes ensambladores. Trabaja como un compilador, produciendo un módulo objeto que puede almacenarse.

Cada sistema computacional tiene comúnmente solo un lenguaje ensamblador a su disposición; así, solo necesita adquirirse un ensamblador.

COMPILADORES:

Un compilador traduce un programa escrito en lenguaje de alto nivel a lenguaje de máquina completamente de una sola vez. Todo lenguaje orientado a los compiladores requiere su propio compilador. Así un programa escrito en lenguaje COBOL necesita un compilador COBOL, no puede traducirse con un compilador FORTRAN. Además, un compilador que funcione con determinada computadora casi seguramente no podrá utilizarse en otra distinta, a menos que exista una cierta compatibilidad entre ellas y el resultado de la compilación también está sometido a consideraciones similares, excepto en los casos de compilación cruzada (se compila en un equipo para que se ejecute en otro específico).

El programa que se escribe en un lenguaje de alto nivel y que se introduce en la computadora se conoce como módulo fuente (o programa fuente).

El programa escrito en lenguaje de máquina que el compilador produce a partir de él es un módulo objeto (o programa objeto).

Antes de que el módulo objeto esté en condiciones de ser ejecutable, por lo común se une a otros módulos objeto que la CPU puede necesitar a fin de procesar el programa. Por ejemplo, la mayoría de las computadoras no pueden calcular directamente raíces cuadradas.

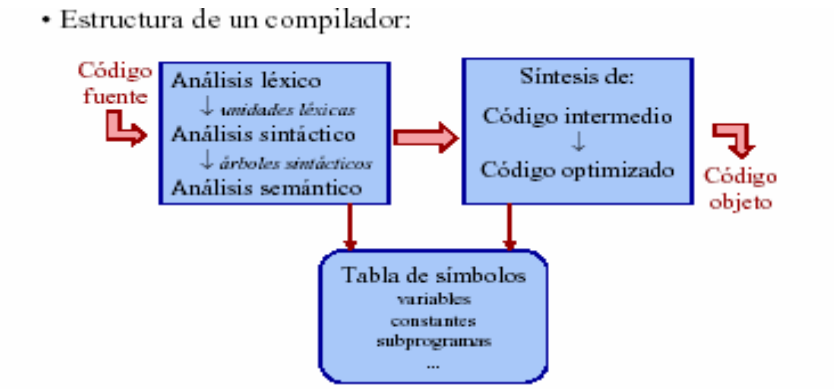
Para hacerlo, se apoyan en pequeños subprogramas, los cuales están almacenados en memoria secundaria en forma de módulos objetos. De este modo, si un programa pide el cálculo de una raíz cuadrada, el sistema operativo unirá la versión del módulo objeto del programa con esta rutina de raíz cuadrada a fin de formar un “paquete ejecutable” para la computadora.

El proceso de unión se conoce como edición de enlace (o etapa de edición de enlace), y el paquete ejecutable que se forma se denomina módulo de carga (o también módulo ejecutable o programa ejecutable).

Los sistemas de computación cuentan con un programa de sistemas especial, denominado editor de enlace, para realizar el enlace de manera automática. Efectivamente, la mayoría de las personas que escriben sus propios programas ni siquiera se dan cuenta de que ocurre la edición de enlace, el sistema operativo se encarga automáticamente de esta operación.

Es el módulo de carga el que la computadora ejecuta en realidad.

Tanto los módulos objeto como los de carga pueden almacenarse en disco para su uso posterior, de modo que la compilación y la edición de enlace no necesitan realizarse cada vez que se ejecute el programa.



INTÉRPRETES: Un intérprete, a diferencia de un compilador, no crea un módulo objeto. Los intérpretes leen, traducen y ejecutan programas fuentes una línea a la vez. De este modo, la traducción al lenguaje de máquina se realiza mientras el programa está siendo ejecutado.

Los intérpretes tienen ventajas y desventajas en relación con los compiladores.

La ventaja principal es que un intérprete requiere mucho menos espacio de almacenamiento. Asimismo, el intérprete no genera un módulo objeto que tenga que ser almacenado.

Muchas versiones del lenguaje BASIC utilizan intérpretes en vez de compiladores, y por esta razón requieren menos almacenamiento que los lenguajes orientados al compilador, como es el caso de COBOL y FORTRAN. Esta es una razón principal por la que el lenguaje BASIC es tan popular en las microcomputadoras, las cuales tienen capacidad limitada de almacenamiento.

La desventaja principal de los intérpretes es que son más lentos y menos eficientes que los compiladores. El programa objeto producido por un compilador se encuentra completamente en lenguaje de máquina, de modo que puede ejecutarse rápidamente.

Los intérpretes, en contraste, traducen cada instrucción inmediatamente antes de ejecutarla, lo cual lleva más tiempo debido a que debe reiterarse este proceso cada vez que se ejecute una instrucción.

Además, el módulo objeto de un programa compilado puede almacenarse en disco, de modo que el programa fuente no tiene que volver a traducirse cada vez que se ejecute el programa; con un intérprete el programa debe ser traducido cada vez que se ejecute.