I

INTRODUCCION A LA PROGRAMACIÓN

Como se sabe, una computadora puede realizar operaciones simples a gran velocidad, pero para que una computadora pueda realizar dichas tareas, se necesita poder indicarle las operaciones que tiene que efectuar. Uno tiene entonces que indicarle el **algoritmo** a seguir.

I.1. ALGORITMOS, PROGRAMAS Y PROGRAMACION.

Un **ALGORITMO** es una *secuencia ordenada, finita e inequívoca de pasos* a seguir para resolver un determinado problema. Es importante el hecho de que sea una secuencia ordenada porque cuando queremos resolver un determinado problema, tenemos que efectuar los pasos en un cierto orden a no ser que queramos obtener un resultado totalmente diferente al esperado. Por ejemplo, en una receta, que es un algoritmo para preparar un determinado platillo, si se altera el orden de los pasos, nunca obtendremos el platillo deseado. Por otro lado, también es importante recalcar que es una secuencia finita de pasos. Un algoritmo debe terminar en algún momento determinado, ya que de nada nos serviría un algoritmo que nos resolviera el problema dentro de mil años.

Existen varias formas de expresar o transmitir un algoritmo, como se discutirá posteriormente con más amplitud. Por ejemplo, una receta se transmite por lo general en forma oral, lo cual puede dar lugar a ambigüedades. Cuando es una computadora la encargada de ejecutar un determinado algoritmo, dicho algoritmo deberá ser expresado en forma de un **PROGRAMA** de computadora, el cual consiste de un conjunto de instrucciones que la computadora pueda entender y posteriormente ejecutar. Para esto uno debe usar un **LENGUAJE DE PROGRAMACION** para escribir un programa. A la actividad de expresar un algoritmo en forma de programa se le denomina **PROGRAMACION**.

A los programas se les denomina empleando el término de **SOFTWARE**, y al equipo físico se le denomina usando el término de **HARDWARE**. Existen ya programas o software previamente desarrollado y del cual pueden hacer uso los usuarios de un sistema de cómputo, pero también gran parte del software, tiene que ser desarrollado por los propios usuarios para sus fines específicos.

En el caso de programas que están destinados a alguna aplicación específica se les conoce como **PAQUETES DE APLICACIONES** como Excel, Word, Mathematica, Matlab, etc. Sin embargo, cuando se desea hacer

algo para lo cual no existe un paquete, uno tiene que escribir sus propios programas para resolver su problema.

Existen además otros programas que son los encargados de proporcionar servicios vitales para que un usuario pueda interactuar con un sistema de cómputo; éstos reciben el nombre de **SOFTWARE DEL SISTEMA**, del cual, un elemento muy importante es el **SISTEMA OPERATIVO**.

El **SISTEMA OPERATIVO** es un conjunto de programas que nos facilitan el uso de los recursos de la máquina. Por ejemplo, mandar a imprimir por una impresora, meter un programa a la computadora, desplegar un programa por la pantalla, etc.

I.2. COMPONENTES DE UNA COMPUTADORA TIPICA.

La **UNIDAD DE CONTROL** es el componente básico de un sistema de cómputo, y con mucho, el más importante, ya que está encargada del control de la operación de todos los demás componentes.

El segundo componente básico es la **UNIDAD ARITMETICO-LOGICA (ALU)** que es en donde, como su nombre lo indica, se efectúan todas las operaciones aritméticas como la suma, resta, etc., y las operaciones lógicas como por ejemplo, la comparación de dos valores.

A la combinación de la Unidad de control y el ALU se le conoce como **CPU (CENTRAL PROCESSOR UNIT)** o **PROCESADOR CENTRAL**. Por ejemplo, tenemos los procesadores de INTEL como el PENTIUM (Pentium II, Pentium III, Pentium IV), los procesadores de Motorola como el 68000, 68030, etc.

Todo sistema de cómputo debe contar además con dispositivos de entrada y salida que le permitan precisamente establecer contacto con el mundo exterior. Tenemos aquí al teclado, monitor, ratón, impresoras, CD-ROM, etc.

Cada computadora tiene una determinada cantidad de almacenamiento interno denominado **MEMORIA PRINCIPAL**. Esta memoria es la que actúa a mayor velocidad. Para que un programa pueda ser ejecutado, éste debe ser almacenado en la memoria principal, la cual está formada por multitud de celdas o posiciones (palabras de memoria) de un determinado número de bits y numeradas en forma consecutiva. A la numeración de las celdas se le denomina dirección de memoria y es mediante esta dirección que se puede acceder en forma directa a cualquiera de ellas. Decimos por ello que la memoria principal es una memoria de acceso directo.

Tenemos aquí dos tipos de memoria interna: la memoria **ROM** (Read Only Memory), en la que solo se permite leer y es permanente, es decir, al apagar la máquina no se pierde la información que ésta tiene. En algunos casos, el contenido de esta memoria está permanentemente grabado desde que se fabricó, en otro se tienen memorias programables de sólo lectura o **PROM** (Programmable Read Only Memory), las cuales pueden borrarse y programarse de nuevo si se cuenta con el equipo indicado.

El segundo tipo de memoria es la **RAM** (**Random Access Memory**) o de acceso aleatorio, la cual, al momento de ser apagado el equipo pierde la información que almacenaba.

Por otro lado está la **MEMORIA SECUNDARIA O MEMORIA EXTERNA**, la cual permite resolver las deficiencias de la memoria principal en cuanto a volatilidad y pequeña capacidad, ya que aunque la memoria interna es muy rápida, no tiene gran capacidad. Tenemos aquí los discos duros, los CD's, los diskettes, etc. la información almacenada en este tipo de memoria no es volátil, es decir, la información es almacenada aquí hasta que el usuario desee borrarla. El precio de esta memoria es notablemente inferior al de la memoria interna.

El CPU opera con las instrucciones de control que proporciona un programador, las cuales, como ya hemos dicho, deben residir en memoria principal. El CPU es el encargado de hacer que los datos necesarios para la ejecución de un programa sean leídos mediante los dispositivos de entrada, almacenados en memoria, llevados y operados en el ALU y mostrar los resultados en algún dispositivo de salida.

El CPU puede entender solamente instrucciones en **lenguaje de máquina**, esto es, en términos de ceros y unos. Cada CPU tiene circuitos especialmente diseñados para ejecutar ciertas instrucciones en particular. La gran dificultad para programar en lenguaje de máquina incentivó el desarrollo de lo que se conoce como **LENGUAJES DE PROGRAMACION**. Surgen los **LENGUAJES DE ALTO NIVEL** (Pascal, Fortran, Basic, etc.), los cuales permiten programar sin necesidad de conocer el funcionamiento interno de la máquina ni su arquitectura, por lo cual, no están sujetos a ninguna máquina en particular, lo que permite su portabilidad. Estos lenguajes están más próximos al usuario y a la notación de sus problemas y resulta por lo tanto mucho más fácil programar en ellos. De aquí que en el caso de un Lenguaje de alto nivel se necesite de un programa traductor que dado un programa en dicho lenguaje, sea capaz de encontrar su equivalente binario, el cual pueda ser entendido por el CPU.

I.3. LENGUAJES DE PROGRAMACION Y TRADUCTORES.

Un lenguaje de programación es un conjunto de símbolos, junto con un conjunto de reglas para combinar dichos símbolos que se usan para expresar programas. Los lenguajes de programación, como cualquier lenguaje, se componen de un léxico (conjunto de símbolos permitidos o vocabulario), una sintaxis (reglas que indican cómo realizar las construcciones del lenguaje), y una semántica (reglas que permiten determinar el significado de cualquier construcción del lenguaje).

Ya hemos dicho que para que una computadora pueda ejecutar un programa escrito en un determinado lenguaje de programación, es necesario que dicho programa sea traducido a un lenguaje que la computadora entienda, el **LENGUAJE DE MAQUINA**, el cual está totalmente apegado a los circuitos de la máquina y muy alejado del lenguaje que los seres humanos utilizan. Aunque dicho lenguaje hace posible hacer programas que utilicen la totalidad de los recursos de la máquina y así obtener programas muy eficientes en cuanto a tiempo de ejecución y uso de memoria, resulta muy difícil programar en él. Por eso se desarrollaron otros tipos de lenguajes como lo vamos a discutir a continuación.

Los lenguajes de programación se pueden clasificar de la siguiente manera, utilizando el criterio de proximidad del lenguaje con la máquina o con el lenguaje natural:

1. **Lenguajes de bajo nivel**: Lenguajes de máquina.
2. **Lenguajes de nivel medio**: Ensambladores y Macroensambladores.
3. **Lenguajes de alto nivel**, como Pascal, Fortran,C, C++, Lisp, Basic, Prolog, etc.

A estos últimos se les puede también clasificar por el tipo de problemas que nos permiten resolver con más facilidad.

1. **Aplicaciones científicas**, en donde predominan operaciones numéricas propias de algoritmos numéricos. Aquí tenemos a Fortran y Pascal, pero particularmente Fortran.
2. **Procesamiento de datos**, como COBOL y SQL.
3. **Tratamiento de textos** como C.
4. **Inteligencia artificial**, como aplicaciones en sistemas expertos, juegos y visión artificial. Aquí tenemos a LISP y PROLOG.
5. **Programación de Sistemas**: Software que permite la interfaz entre el hardware y el usuario. Tenemos a ADA, MODULA-2 y C

Otra clasificación sería por el estilo de programación que fomentan.

1. **Lenguajes imperativos o procedurales**. Estos establecen cómo debe ejecutarse una tarea, dividiéndola en partes y especificando las subtareas asociadas. Se fundamentan en el uso de variables para almacenar valores y el uso de instrucciones para indicar operaciones a realizar con los datos. La mayoría de los lenguajes de alto ni-

vel son de este tipo: Fortran, Pascal, Basic, etc.

2. **Declarativos**. Los programas se construyen mediante descripciones de funciones o expresiones lógicas que indican las relaciones entre determinadas estructuras de datos (PROLOG).

3. **Lenguajes orientados a Objetos**. Se centran más en los datos y su estructura. Un programa consiste de descripciones de unidades denominadas objetos que encapsulan a los datos y las operaciones que actúan sobre ellos (C++).

4. **Lenguajes orientados al problema**. Diseñados para problemas específicos. Son generadores de aplicaciones que permitan automatizar la tarea de desarrollo de software de aplicaciones.

Se hablará entonces ahora de los traductores para los diferentes tipos de lenguajes de programación, de acuerdo con su proximidad al lenguaje de máquina o al lenguaje natural.

I.3.1. ENSAMBLADORES.

El lenguaje **ENSAMBLADOR** es un primer intento de sustituir el lenguaje de máquina por uno más cercano a nosotros, los humanos. Cuando se asocia un mnemónico a una instrucción de máquina, tenemos lo que se conoce como **LENGUAJE ENSAMBLADOR**. En un lenguaje ensamblador, el direccionamiento también es simbólico, es decir, en lugar de utilizar direcciones binarias absolutas para accesar a los datos, éstos pueden ser identificados usando nombres como SUMA; X, A, B, etc. Además se permite el uso de comentarios, lo cual hace que los programas sean más entendibles.

Un programa llamado **ENSAMBLADOR** traduce las instrucciones en lenguaje ensamblador a lenguaje de máquina. A la entrada de un programa ensamblador se le conoce como **programa fuente** y a la salida como **programa objeto**.

Sin embargo, este tipo de lenguajes presenta la mayoría de los inconvenientes del lenguaje de máquina, ya que su conjunto de instrucciones es muy reducido y rígido. No hay portabilidad ya que hay una fuerte dependencia con el hardware de la computadora. La ventaja, como ya hemos dicho, es que permite el uso óptimo de los recursos de la máquina.

Para resolver las limitaciones de este tipo de lenguajes desarrollan unos ensambladores especiales, los macroensambladores.

I.3.2. MACROENSAMBLADORES.

Al hacer un programa en lenguaje Ensamblador se encuentra uno a veces con la necesidad de repetir algunas partes del código. Se llaman **MACROINSTRUCCIONES (o MACROS)** a las abreviaturas para un grupo de instrucciones. Una sola instrucción representa un bloque de código.

Un **MACROENSAMBLADOR** es un programa que traduce un lenguaje de macroinstrucciones a lenguaje de máquina.

Con el fin de hacer la programación independiente de la máquina en la que se esté programando, surgen, como ya se ha dicho, los lenguajes de alto nivel. Estos usan frases relativamente fáciles de entender y están más apegados al lenguaje de los problemas que se quiere resolver.

Un lenguaje de alto nivel, como ya se ha dicho anteriormente, es independiente de la arquitectura de la máquina y entonces, al igual que ocurre incluso con los lenguajes ensambladores y macroensambladores, se necesita de un traductor que traduzca las instrucciones en un lenguaje de alto nivel a lenguaje de máquina. En este caso, una instrucción en lenguaje de alto nivel, da lugar a varias instrucciones en lenguaje de máquina. Se suele también incluir instrucciones de uso frecuente como por ejemplo funciones matemáticas de uso común (seno, coseno, logaritmo, etc.). Existe una gran cantidad de lenguajes de alto nivel y para su traducción se requiere de compiladores o intérpretes.

I.3.3. COMPILADORES

Es un programa que acepta un programa fuente en un lenguaje de

alto nivel y produce su correspondiente programa objeto (programa ya en lenguaje de máquina).

Algunos compiladores traducen sólo programas completos, mientras que otros traducen partes de un programa. Se puede, en este caso, dividir un programa en **MODULOS**, donde un módulo es la unidad más pequeña de software que resuelve un subproblema del problema general que se quiere resolver. El programa principal controla todo lo que sucede y los submódulos o subprogramas (como entrada y salida, manipulación de datos, control de otros módulos, combinación de los anteriores) se ejecutan, devolviendo el control al programa principal. Cada submódulo es independiente y solamente tiene acceso directo al módulo al que llama y sus submódulos. Cada módulo puede compilarse (e incluso probarse) de manera independiente.

Se necesita entonces de un **LIGADOR** que una los módulos traducidos en un sólo programa.

I.3.4. INTERPRETES.

Un lenguaje de alto nivel, además de ser compilado, puede ser interpretado. Un **INTERPRETE** es un programa que traduce, al igual que un compilador, programas escritos en un lenguaje de alto nivel a lenguaje de máquina; sin embargo, en este caso no existe independencia entre la fase de traducción y la de ejecución. Un intérprete traduce cada instrucción en un lenguaje de alto nivel a lenguaje de máquina e inmediatamente se ejecuta; a continuación se ejecuta la siguiente instrucción, etc. Como ejemplo tenemos los intérpretes de BASIC.

Un intérprete no traduce todo el programa en un solo paso, sino que traduce y ejecuta cada instrucción antes de traducir y ejecutar la siguiente.

I.3.5. CARGADORES.

Un **CARGADOR** es un programa que carga un programa objeto a memoria principal y lo prepara para su ejecución.

I.4. SISTEMAS OPERATIVOS.

Un **SISTEMA OPERATIVO** es un conjunto de programas que permiten utilizar los recursos de la máquina. Esto es, sirve como un enlace entre el hardware y el usuario.

Como ejemplos de sistemas operativos tenemos MS-DOS, OS/2, UNIX (y sus diferentes versiones como SOLARIS, IRIX, LINUX, etc.), WINDOWS-NT, VMS (Vax), así como los sistemas operativos de red y sistemas operativos distribuidos.

Los sistemas operativos de red son una ampliación de los sistemas operativos convencionales que permiten el control de una red de computadoras. Disponen de programas de control de interfaz con la red y permiten establecer una sesión de trabajo con un sistema remoto, transferir archivos entre computadoras, intercambiar correo, etc.

Los sistemas operativos distribuidos controlan el procesamiento distribuido. Este implica la conexión en paralelo de varias computadoras ejecutando funciones concurrentemente y comunicándose entre si. Estos resultan muy complejos.

Un sistema operativo debe ser

1. Eficiente, es decir, no debe desperdiciar tiempo útil y debe realizar sus funciones de una manera rápida.

2. Fiable, ya que un fallo de él, puede causar que el sistema se "caiga".

3. Deben ser de tamaño pequeño.

Un sistema operativo debe contar con programas de apoyo que permitan realizar operaciones como:

a) EDITAR.

Durante el desarrollo de un programa, por lo general, resulta necesario efectuar correcciones, agregar módulos, etc. Para evitar el tener que volver a teclear grandes porciones se acostumbra mantener el programa en almacenamiento secundario. Al programa que se utiliza para meter un programa por primera vez, y en general un texto cualquiera como manuales o cartas, y realizar correcciones, recibe el nombre de **EDITOR**.

Un editor nos permite efectuar operaciones como:

1. Eliminar partes.
2. Reemplazar partes.
3. Insertar partes.

b) TRANSFERIR INFORMACION.

Un sistema operativo debe ser capaz de permitir transferir información de memoria principal a memoria secundaria, y viceversa. También debe poder permitir sacar respaldos de discos, etc.

La información en un disco está organizada mediante **ARCHIVOS**. Un archivo es una colección de información relacionada entre sí y es comparable a un folder en un archivero. Todos los programas, textos, imágenes, etc., en un disco, residen en archivos.

Un sistema operativo deber ser capaz de desplegar los nombres de los archivos almacenados en un disco, así como sus contenidos. Debe poder permitir el borrado de archivos que ya no se utilicen. También debe permitir mandar a imprimir el contenido de un archivo, etc.

c) EJECUTAR PROGRAMAS.

Mediante el sistema operativo debe ser posible el ejecutar un programa que ya se encuentre traducido a lenguaje de máquina. El programa, como ya hemos dicho, debe ser cargado a memoria principal antes de ser ejecutado.

Aquí se va a estudiar el sistema operativo MS-DOS y además LINUX, que son los sistemas operativos con los que se va a trabajar.

Otras tareas de los sistemas operativos son por ejemplo el mantenimiento de una contabilidad del gasto de recursos que realiza cada uno de los usuarios. Claro que esto tiene sentido cuando un sistema tiene varios usuarios(sistemas operativos multiusuario). Generalmente se desea sobre todo contabilizar el tiempo de procesador consumido por cada proceso.

Existen entonces además de los **sistemas operativos uniusuario** tradicionales, otras **categorías de sistemas operativos**:

1. **Sistemas operativos multitarea** (o multiprogramados), los cuales son capaces de ejecutar más de un programa a la vez. Estos se basan en técnicas de multiprogramación y son los más extendidos en la actualidad.

2. **Sistemas operativos multiusuario**, los cuales son sistemas que permiten que más de un usuario accese al sistema. Naturalmente dicho sistema debe ser también multitarea ya que cada usuario podrá ejecutar varios programas a la vez. UNIX es un ejemplo de este tipo de sistemas.

3. **Sistemas operativos multiproceso**: existen sistemas que tienen dos o más procesadores interconectados, trabajando simultáneamente. En este caso, el sistema operativo debe ser capaz de administrar el reparto del trabajo entre los distintos procesadores para sacar provecho del paralelismo existente. (LINUX, UNIX y WINDOWS-NT).