Centro Asociado Palma de Mallorca

Arquitectura de Ordenadores

Tutor: Antonio Rivero Cuesta

Unidad Didáctica 2

Estructura de un Computador

Tema 6

Programación de Ordenadores

Conceptos de Programa de Ordenador y Lenguaje de Programación

Un programa de ordenador es una secuencia de instrucciones que el ordenador ejecuta una tras otra.

Dichas instrucciones se almacenan en la memoria y deben ser escritas previamente por un ser humano.

A éste se le denomina programador del ordenador.

El programador escribe el programa en lo que se denomina lenguaje de programación.

Un programa maneja datos, es decir información, y realiza manipulaciones con la misma.

Tanto los datos como las instrucciones se encuentran almacenados en la memoria del ordenador.

Un ordenador sólo entiende instrucciones codificadas en forma de secuencias de unos y ceros.

Estos unos y ceros se traducen finalmente en señales eléctricas que activan los circuitos del computador.

El conjunto de instrucciones codificadas en unos y ceros que el ordenador entiende, se denomina ***lenguaje máquina***.

Este lenguaje está próximo al ordenador pero muy lejano de los lenguajes que utilizamos los humanos.

Un lenguaje de programación que está lejos del lenguaje humano se dice que es un ***lenguaje de bajo nivel***.

Por lo tanto el lenguaje máquina es un lenguaje de bajo nivel.

De hecho el nivel más bajo posible.

Para facilitar la tarea de programación de los ordenadores se han inventado lenguajes, más asequibles al ser humano que el lenguaje máquina.

Éstos se denominan **lenguajes de alto nivel**.

El Código Máquina

Como veíamos anteriormente, el lenguaje máquina es el único que entiende realmente la CPU del ordenador.

En este lenguaje los programas se expresan como secuencias de unos y ceros.

Un dato muy importante a tener en cuenta es que cuando se escribe en código ensamblador hay que conocer de antemano las posiciones de memoria en las que se aloja el código del programa, así como las posiciones en las que van los datos.

Antes de manipular un dato hay que buscarlo en la memoria, siendo necesario por tanto saber dónde está.

Análogamente ocurre cuando hay que guardar el dato.

Asimismo los programas contienen instrucciones de salto a otras instrucciones, siendo necesario conocer en qué posición de memoria se encuentran éstas.

Todo esto configura una situación que hace prácticamente inviable la programación en lenguaje máquina salvo para tareas muy próximas al hardware.

El Lenguaje Ensamblador

Un ordenador tendrá su juego de instrucciones en código binario.

El cual constituye su lenguaje máquina.

En vez de utilizar directamente los unos y ceros podemos asignar a cada instrucción un nombre.

Este nombre está compuesto de letras y números o dicho más técnicamente se escribe en código alfanumérico.

Por ejemplo MOVE se utiliza para mover información de una posición a otra de la memoria.

El programador, en lugar de escribir código binario escribe los nombres nemotécnicos de las instrucciones, lo cual es mucho más manejable.

El lenguaje así construido se denomina **lenguaje ensamblador**.

Ventajas e Inconvenientes del Lenguaje Ensamblador Frente A los Lenguajes de Alto Nivel

Ventajas

Un programa en ensamblador es más eficiente.

Es más conveniente que otros lenguajes para programar tareas muy específicas tales como control de periféricos, gestión de la memoria etc., ya que se tiene control de las señales que circulan prácticamente a nivel de bit.

Inconvenientes

El lenguaje ensamblador está tan próximo al hardware del ordenador que para realizar cualquier programa es necesario conocer la arquitectura del mismo.

En un programa en ensamblador hay que saber en qué posiciones de memoria van las instrucciones y los datos.

La programación de cualquier tarea debe ser descompuesta en multitud de pasos elementales siendo necesario una gran cantidad de tiempo para desarrollar los programas.

Los programas son difíciles de leer y entender y en consecuencia de mantener.

Lenguajes Macroensambladores

Para aumentar la eficiencia en la programación se recurre a crear instrucciones denominadas macroinstrucciones que agrupen a varias instrucciones de código máquina.

La ejecución de una macroinstrucción se convierte en la ejecución de varias instrucciones de lenguaje máquina.

El lenguaje así construido se denomina macroensamblador.

Lenguajes de Alto Nivel

Los lenguajes ensambladores así como los macroensambladores son demasiado primitivos para el ser humano.

Los principales problemas que enfrenta el programador son:

El programador en lugar de estar ocupado esencialmente en el problema que desea resolver está dedicado a resolver multitud de detalles relacionados con la arquitectura del computador.

Por otra parte como las instrucciones en ensamblador y macroensamblador son poco potentes, se requiere

gran cantidad de ellas para ejecutar cualquier tarea que tenga significado en el contexto humano.

A esto se añade que el programa desarrollado sólo funciona en el ordenador en el que se ha escrito o bien en otro que tenga un hardware idéntico.

Toda esta complejidad hace que si se produce algún error, sea muy difícil localizarlo y arreglarlo.

Finalmente todos estos problemas repercuten en el tiempo de desarrollo de los programas y en el coste de los mismos.

Para solucionar todos estos problemas se han desarrollado los lenguajes de alto nivel.

Los lenguajes de alto nivel se pueden clasificar atendiendo a varios criterios.

Si los clasificamos según el tipo de aplicación que se dará al programa desarrollado tenemos la siguiente clasificación:

Lenguajes Aplicaciones Científicas y Técnicas

Se trata de lenguajes que facilitan la programación de algoritmos.

Están orientados al cálculo matemático.

Los programas desarrollados son rápidos pudiendo realizar muchas operaciones por segundo y ocupan poco espacio en memoria.

A cambio suelen dar pocas facilidades al programador para crear una interface amigable con el usuario, dibujar gráficos o manejar ficheros de datos.

En esta categoría entran los lenguajes C, Pascal o Fortran.

Lenguajes Aplicaciones de Gestión

Son lenguajes especializados en manejar gran cantidad de información que se encuentra almacenada en el disco normalmente en estructuras llamadas bases de datos.

Estos programas permiten desarrollar eficientemente, aplicaciones de gestión de empresas tales como programas de contabilidad, nóminas, facturación, gestión de almacenes, etc.

En esta categoría entran lenguajes como los ya clásicos COBOL, SQL, ORACLE, dBase, Visual Objects, etc.

Lenguajes para Aplicaciones de Software de Base y Utilidades

Denominamos software de base a programas tales como sistemas operativos, compiladores, ensambladores, programas de comunicaciones etc.

Realizan tareas como el control primario del ordenador y la gestión del hardware del mismo, la traducción de programas a código máquina, la gestión de transferencia de información entre ordenadores a través de redes de datos etc.

Como utilidades mencionaremos aquellas que complementan al sistema operativo propiamente dicho, tales como compresores de disco, interfases gráficas con el usuario, así como también otro tipo de programas que es un poco discutible que entren en esta clasificación, tales como procesadores de texto, hojas de cálculo, bases de datos, etc.

Todos estas aplicaciones requieren que el lenguaje de programación permita controlar el ordenador a bajo nivel produciendo programas rápidos y pequeños en memoria.

Además deben tener las características de un lenguaje de alto nivel, para poder desarrollar el código en poco tiempo.

En esta categoría entran el C que es el más utilizado y también PASCAL y MODULA.

Lenguajes para Aplicaciones de Inteligencia Artificial

Facilitan la programación de aplicaciones tales como sistemas expertos.

Son útiles para programar diversas técnicas de representación del conocimiento utilizadas en inteligencia artificial.

Los lenguajes más utilizados son PROLOG y LISP.

Atendiendo a la filosofía de programación podemos clasificar los lenguajes en:

Imperativos o Procedurales

El programa indica que pasos debe seguir el ordenador para realizar una tarea dada.

Es decir describe el procedimiento que utilizará el ordenador para realizar la tarea.

Los programas se dividen en bloques de código llamados funciones o procedimientos.

Las instrucciones operan sobre variables o datos que se encuentran en la memoria. La mayoría de los lenguajes pertenecen a esta categoría. Los más importantes son: C, Pascal, Fortran, ADA, MODULA2.

Orientados a Objetos

Permiten aplicar la filosofía de orientación a objetos. Según esta filosofía un objeto es una entidad que consta de métodos y propiedades.

En programación los métodos son funciones y procedimientos y la propiedades son variables.

Cualquier problema se puede representar mediante este tipo de objetos.

El objeto más general se denomina Universo y contiene a todos los demás.

Los objetos se relacionan unos con otros mediante diversos mecanismos.

Los programas realizados con esta filosofía o metodología se desarrollan en menos tiempo, son más fiables y más fáciles de mantener.

Además es más fácil reutilizar el código que ha sido desarrollado según esta filosofía.

En la práctica los lenguajes orientados a objetos son evoluciones de los lenguajes imperativos.

Los más utilizados son: C++, Objects Pascal, Visual Basic y Java.

Lenguajes Declarativos

El proceso que sigue el ordenador para realizar la tarea deseada no aparece explícitamente en el programa.

Por el contrario el programa describe el problema que se desea resolver, mediante relaciones entre funciones o entre estructuras de datos.

Los lenguajes más populares de este tipo son LISP y PROLOG.

Lenguajes Orientados al Problema

Han sido diseñados para resolver con eficiencia determinados tipos de problemas.

Cabe destacar los lenguajes de gestión de bases de datos que permiten desarrollar de forma eficiente programas especializados en gestionar gran cantidad de información. Entre estos el más importante es el COBOL.

Otra clasificación habitual en los lenguajes de programación es por generaciones.

Según esta clasificación tenemos:

Lenguajes de primera generación. Son los lenguajes máquina.

Lenguajes de segunda generación. Son los lenguajes ensambladores y macroensambladores.

Lenguajes de tercera generación: Son los lenguajes imperativos más utilizados en la actualidad. C, PASCAL, MODULA, ADA, etc.

Lenguajes de cuarta generación. Son lenguajes acompañados de facilidades para la programación.

Estas facilidades ahorran trabajo al programador de tal forma que parte del programa se genera automáticamente.

Entre estos lenguajes se encuentran algunos de los orientados al problema.

Lenguajes de quinta generación. Se elimina más la necesidad de escribir instrucciones.

Podemos situar en esta categoría a los lenguajes orientados a inteligencia artificial.

Algunos de estos lenguajes consisten en una colección de procedimientos que el programador debe seguir para indicar al ordenador qué es lo que debe hacer, pero el programador no escribe ninguna instrucción propiamente dicha.

Entre estos lenguajes se encuentra el MAGIC.

Entornos de Programación

Los entornos de programación son programas que proveen de todas las herramientas necesarias para que el programador desarrolle un programa.

Las herramientas más importantes son:

- El editor.
- Los programas traductores.
- El depurador.

Las facilidades que aportan los citados entornos son:

Edición del programa. El programador escribe las instrucciones como si estuviera en una máquina de escribir.

El programa que permite hacer esto se denomina editor.

Además de permitir escribir, el editor puede aportar otras facilidades tales como:

Análisis del código sobre la marcha, indicando algunos errores según el programador está mecanografiando.

Estos errores son fundamentalmente sintácticos.

El editor avisa al programador de los errores cometidos según éste va escribiendo.

Ayuda en tiempo real al programador. El editor presenta información al programador con la sintaxis de la instrucción que éste ha comenzado a teclear de tal manera que pulsando una tecla el programador completa la palabra en curso sin tener que escribirla completamente.

El editor puede formar parte del entorno de desarrollo o bien ser una herramienta aparte, que el usuario arranca por separado.

En los sistemas modernos tanto el editor como el resto de las herramientas suelen estar integradas en un mismo programa.

El programa escrito en el editor se denomina código fuente.

El editor permite guardar este programa en un fichero que se denomina fichero fuente.

Este fichero será procesado posteriormente por otras herramientas del entorno de desarrollo.

Programas traductores. Realizan la traducción del código fuente a código máquina convirtiéndolo en lo que se denomina un programa ejecutable.

Esto puede realizarse de varias maneras sobre las que volveremos posteriormente.

Normalmente también forman parte del entorno de programación.

Herramientas de depuración del programa en ejecución. Una vez que el programa ha sido escrito y se ha convertido a un ejecutable, hay que probarlo pues puede haber errores.

Normalmente cuando un programa con errores se ejecuta, los efectos sobre el ordenador son desastrosos pudiendo incluso ser necesario rearrancar el mismo.

Los entornos de desarrollo actuales protegen al resto de los programas en ejecución, de los efectos de los programas con errores.

Además facilitan la detección de éstos por el programador permitiendo que éste ejecute el programa en desarrollo instrucción a instrucción y analice al mismo tiempo el estado de todas las variables y otros elementos importantes.

Programas Traductores

La traducción del programa fuente a ejecutable o código máquina requiere varias etapas y se realiza de forma automática por uno o varios programas denominados traductores.

Hay dos tipos de procedimientos para realizar la traducción: interpretar instrucción a instrucción el código fuente o bien compilar el programa y enlazarlo.

Programas Intérpretes

Convierten a código máquina las instrucciones y las ejecutan una a una.

El intérprete no trabaja sobre el programa en su conjunto sino que traduce una instrucción y la ejecuta.

A continuación continúa con la siguiente instrucción y así sucesivamente.

El programa nunca existe como tal en forma ejecutable.

Sólo existe como ejecutable una instrucción del mismo en cada momento.

Los intérpretes permiten experimentar el resultado de nuestro programa de forma inmediata pues la traducción y ejecución de una instrucción se realiza muy rápidamente.

Por esto son muy cómodos para desarrollar rápidamente programas.

A cambio el intérprete no puede optimizar el código del programa puesto que para esto se necesita analizar el mismo de forma global.

Compiladores

Aquellos programas que no se ejecutan bajo intérprete requieren de un proceso de conversión a código máquina o código ejecutable, que comprenda a todo el programa en su conjunto.

Los compiladores son programas que traducen el código fuente a código objeto.

El código objeto es código máquina, pero normalmente las direcciones de las variables en la memoria requieren un ajuste adicional para que el programa pueda ejecutarse.

Por tanto el código objeto no es ejecutable.

El compilador no determina en que posiciones de memoria se cargará el programa ejecutable.

Téngase en cuenta además que normalmente las zonas de la memoria donde se encuentra el código y los datos son diferentes aunque están contiguas.

Un hecho que justifica esto es que los programas fuente a partir de un cierto tamaño están normalmente organizados en varios ficheros o módulos.

El programador no escribe todas las instrucciones en un solo fichero ya que este podría ser demasiado grande.

Estos ficheros están interrelacionados de tal forma que pueden compartir algunas variables entre ellos, así como pueden utilizarse en unos ficheros funciones cuyo código se encuentra escrito en otros.

Todos los lenguajes disponen de librerías que contienen funciones que podemos utilizar.

Estas funciones son de uso tan común que normalmente se proporcionan con el entorno de desarrollo para que el programador no tenga que escribirlas.

Las librerías se encuentran en código objeto y lógicamente no pueden tener asignadas las direcciones

de las variables ni de las instrucciones puesto que están pensadas para formar parte de cualquier programa.

Así pues después del proceso de compilación tenemos nuestro programa dividido en uno o varios módulos objeto.

Estos módulos pueden haber sido escritos por nosotros o bien ser librerías.

Estos módulos se combinan finalmente para dar lugar al programa ejecutable.

Si en los ficheros fuente hay errores, los compiladores no generan el código objeto, en lugar de esto generan

mensajes de ayuda para que el programador corrija los errores.

Los compiladores no pueden en general detectar todos los posibles errores que los programadores pueden cometer.

El Análisis Lexicográfico

Para detectar errores el compilador somete al fichero fuente a un análisis lexicográfico.

Durante este proceso el compilador recorre el fichero fuente separando las diferentes unidades del lenguaje, también denominadas tokens, que componen el programa.

Estos tokens son las palabras reservadas, los nombres de las variables, los operadores y las constantes.

En esta fase del análisis se detectan varios tipos de errores:

- Utilizar nombres ilegales para las variables o las funciones y procedimientos.
- Utilizar incorrectamente palabras reservadas.
- Utilizar incorrectamente constantes.

En esta fase el compilador crea una tabla de símbolos.

Ésta es una lista de todos los nombres de variables, funciones y procedimientos que aparecen en el programa fuente.

Posteriormente se asignará a cada elemento de esta lista una dirección de memoria.

El Análisis Sintáctico

Un lenguaje de programación especifica un conjunto de reglas sintácticas que deben ser respetadas.

Una de las razones para que existan estas reglas es precisamente que el proceso de traducción a código máquina pueda ser efectuado de forma automática.

Cuando el programador no respeta estas reglas el compilador no puede traducir.

El analizador sintáctico (o parser) del compilador recibe los tokens y busca posibles errores. Éstos pueden ser:

- La estructura de la frase no es correcta.
- No se reconoce algún símbolo.

El Análisis Semántico

Detecta incoherencias en el programa.

Por ejemplo un bloque de código puede no ser accesible pues el flujo de ejecución de instrucciones nunca pasará por él.

Una instrucción sintácticamente correcta puede no tener ningún significado y por lo tanto no podrá ser traducida a un bloque de código máquina con el mismo significado.

También puede ocurrir que se programen operaciones incoherentes.

Optimización del Código

Finalmente el compilador intenta optimizar el código del programa.

Para ello elimina todas las instrucciones que se han generado en código máquina y son superfluas.

Téngase en cuenta que al ser automático el proceso de traducción se generan instrucciones que no son relevantes y pueden ser eliminadas.

El compilador también toma decisiones para adaptar el código a la arquitectura de la CPU.

El programa optimizado debe ocupar menos memoria y correr más deprisa.

El Montador de Enlaces

Una vez que los diferentes módulos fuente del programa han sido compilados hay que combinar los módulos objeto correspondientes en un solo programa.

La herramienta o utilidad que realiza esta función es el montador de enlaces, más conocido como enlazador o linkador.

El procedimiento básico para conseguirlo es disponer los módulos objeto uno tras otro en un solo fichero y a continuación dar direcciones a todos los símbolos: a las variables se les asigna su dirección en memoria y todas las referencias a las distintas variables tienen

que apuntar a las correspondientes direcciones de memoria.

Las funciones obtienen una posición de memoria también.

Cada vez que en una instrucción del programa se produce una llamada a una función, se coloca una instrucción de salto a la posición de memoria en la que se encuentra ubicado el código de la función.

Previamente se ha establecido un mecanismo para pasar los argumentos a la función.

Análogamente ocurre con las llamadas a procedimientos.

Éstos, a diferencia de las funciones, no devuelven ningún valor por lo que el mecanismo de salida del procedimiento es ligeramente más sencillo que el de las funciones.

Nótese que cuando se llama a un procedimiento o función el código del mismo ni siquiera tiene que encontrarse en el mismo fichero objeto que la instrucción de llamada.

El montador de enlaces repasa todo el programa asignando valores a todas las referencias de la tabla de símbolos hasta que finalmente todas ellas han sido resueltas.

Si se hace alguna llamada a una función cuyo código no existe el montador de enlaces genera un error y el fichero ejecutable no se produce.

Nótese que hay una diferencia entre aquellos sistemas o computadores que ejecutan varios programas de usuario concurrentemente y los que sólo tienen un programa de usuario en la memoria para ejecución.

Cuando el sistema soporta varios programas, que el usuario puede cargar a su antojo, al mismo tiempo en la memoria, las direcciones de los símbolos no pueden ser fijas pues se producirían conflictos en el uso de la memoria entre unos programas y otros.

En este caso se asignan a los símbolos direcciones relativas respecto de la dirección que el sistema operativo le da al programa en el momento de ser cargado en memoria.

Se dice entonces que el programa es reubicable.

También se dice que sistema es multitarea.

La mayoría de los computadores actuales son multitarea, incluidos los populares PCs con sistema operativo Windows.

No obstante existen multitud de sistemas basados en microprocesador que están dedicados a una tarea específica y sólo ejecutan un programa.

En este caso el código no necesita ser reubicable y el enlazador asigna las direcciones absolutas al programa.

Ejemplos de estos sistemas se encuentran en los automóviles, ascensores, sistemas de control de plantas energéticas sistemas médicos etc.

El número de aplicaciones es inacabable.

Ejemplos de Lenguajes de Alto Nivel

A continuación daremos una breve semblanza de los lenguajes de alto nivel más populares:

- FORTRAN.
- COBOL.
- BASIC.
- PASCAL.
- Lenguaje C.