1. INTRODUCCIÓN.

La mayoría de las actividades que las personas hacen en cualquier sector de una empresa para tratar los datos puede descomponerse en cuatro pasos:

1) Captar datos
2) Almacenar datos
3) Procesar datos
4) Presentar los datos procesados.

Estos pasos son cada vez más complejos y sobre todo repetitivos, lo que hizo que se investigase para automatizarlos, agilizarlos y conseguir que se produzcan cada vez menos errores en dicho procesamiento.

El ordenador va ser la máquina encargada de realizar dicha automatización.

Ejemplo: Cuenta el número de filas del aula y el número de ordenadores ¿Cuántos hay en total? Divídelo en el siguiente proceso:

- Toma de datos: filas y ordenadores por fila.

- Almacén de datos: papel o mi cabeza.

- Proceso de datos: realizar la multiplicación.

- Presentar los datos: decirlos o escribirlos.

2. ALGORITMO.

¿Cómo se hace el procesamiento de información?:

- Se parte de unos datos iniciales llamados datos de entrada o simplemente ENTRADA.
- Los datos se ALMACENAN en el interior del ordenador.
- Los datos almacenados se manipulan, PROCESO, utilizando un conjunto de reglas o instrucciones, llamado programa, con el fin de obtener un determinado resultado o resultados.
- Los resultados obtenidos se denomina SALIDA.

El programa a su vez está también almacenado internamente en el ordenador y es ejecutado por éste de forma automática y secuencial cuando así se lo indiquemos.

ALGORITMO: es una estrategia para resolver un problema en un número finito de pasos.

También se define como un conjunto ordenado y finito de reglas u órdenes que ejecutadas tal y como se escriben, resuelven el problema planteado con independencia de los datos o circunstancias del problema.

El algoritmo no debe tener ambigüedades, y por tanto debe contemplar todas y cada una de las situaciones diferentes que puedan presentarse en distintas resoluciones del problema.

La resolución de un problema puede hacerse mediante diferentes algoritmos. Pero no todos tienen la misma calidad.

Los **parámetros para medir la calidad de los algoritmos** son:

- Mínima ocupación de memoria.
- Mínimo tiempo de ejecución.
- Legibilidad, es decir, debe estar escrito de forma que se facilite su lectura y comprensión.
- Portabilidad, es decir el algoritmo debe estar concebido de forma que se pueda codificar con facilidad en distintos lenguajes de programación.
- Modificabilidad, rara vez un programa no debe adaptarse a nuevas exigencias del usuario, por ello los programas deben estar escritos de forma que contemplen esta circunstancia.

En conclusión, todo algoritmo debe tener las siguientes características:

- Un punto de inicio.
- Estar **bien definido**, es decir hacer lo que se desea que haga. Si se parte de los mismos datos, siempre se ha de obtener el mismo resultado.
- Tener en cuenta todas las posibles situaciones que puedan plantearse.
- Ser **preciso**, indicando el orden de realización de cada uno de sus pasos.
- Fácil de leer e interpretar.
- Ser **finito**. Al seguir un algoritmo este debe concluir en algún momento. Tiene un número finito de pasos.
- Terminar en un solo punto.
- Estar diseñado de forma que sea independiente del lenguaje de programación en que vaya a ser codificado.

Ejemplos de algoritmos.

Ejemplo 1: Un cliente realiza una compra en una tienda con su tarjeta VISA, la tienda examina en el banco correspondiente si el cliente es solvente, en cuyo caso acepta la compra y en caso contrario rechaza la compra.

```
Entorno
      número, banco, cliente, importe, cuenta
Fin_Entorno
```

1. Inicio.
    2. Leer el número de la VISA.
    3. Llamar al banco.
    4. Comprobar con el banco si el cliente es solvente.
    5. Si el cliente no es solvente se le rechaza la compra y vamos al paso 9.
    6. Si el cliente es solvente se acepta la compra por tarjeta VISA.
    7. Se carga el importe de la compra en la cuenta del cliente.
    8. Se abona el importe de la compra en la cuenta de la tienda.
9. Fin.

Ejemplo 2: Realizar el algoritmo que describa el proceso de ponerse los zapatos.

Solución incorrecta por no cumplir las características principales de los algoritmos:

Entorno
    Zapato, cordones
Fin_Entorno

1. Inicio.
    2. Coger los zapatos.
    3. Atarse los cordones del zapato izquierdo.
    4. Sentarse en una silla.
    5. Meterse un zapato en el pie derecho.
    6. Atarse los cordones del zapato izquierdo.
    7. Meterse el otro zapato en el pie izquierdo.
    8. Atarse los cordones del zapato izquierdo.
    9. Si los cordones de los zapatos están anudados se desatarán.
    10. Volver al paso 2.
11. Fin.

Solución correcta:

Entorno
    zapato, cordones
Fin_Entorno

1. Inicio.
    2. Coger los zapatos.
    3. Si los cordones de los zapatos estan anudados se desatarán.
    4. Sentarse en una silla.
    5. Meterse el zapato derecho en el pie derecho.
    6. Atarse los cordones del zapato derecho.
    7. Meterse el zapato izquierdo en el pie izquierdo.
    8. Atarse los cordones del zapato izquierdo.
9. Fin.

Ejemplo 3 (Estructura secuencial): Hervir unas patatas.

Entorno
recipiente, patatas, agua, sal, fuego
Fin_Entorno

1. Inicio.
    2. Coger unas patatas de la bolsa.
    3. Pelar las patatas.
    4. Lavar las patatas.
    5. Poner agua en un recipiente.
    6. Introducir las patatas en el recipiente.
    7. Poner el recipiente al fuego.
    8. Esperar la ebullición del agua.
    9. Esperar quince minutos.
    10. Apagar el fuego.
    11. Retirar las patatas del agua.
12. Fin.

Ejemplo 4 (Estructura condicional): Elección de la prenda de abrigo.

Entorno
abrigo, gabardina, ventana
Fin_Entorno

1. Inicio.
    2. Mirar por la ventana.
    3. Si llueve coger la gabardina; si no coger el abrigo,
4. Fin.

Ejemplo 5 (Estructura repetitiva): Trabajo del empleado que abre una tienda de alquiler de vídeos. El empleado tiene que abrir el cajón donde están las películas devueltas (siempre habrá alguna) por los socios y colocar una a una en su lugar.

Entorno
Cajón, cintas, estantes
Fin_Entorno

1. Inicio.
    2. Abrir el cajón.
    3. Coger una cinta
    4. Colocar la cinta en su lugar.
    5. Mirar si quedan más cintas en el cajón.
    6. Si hay más, volver a 3 y repetir 4,5 y si no hay más
    7. Cerrar el cajón
8. Fin

3. ELEMENTOS DE UN SISTEMA INFORMÁTICO

A la hora de diseñar un sistema informático se tiene en cuenta:

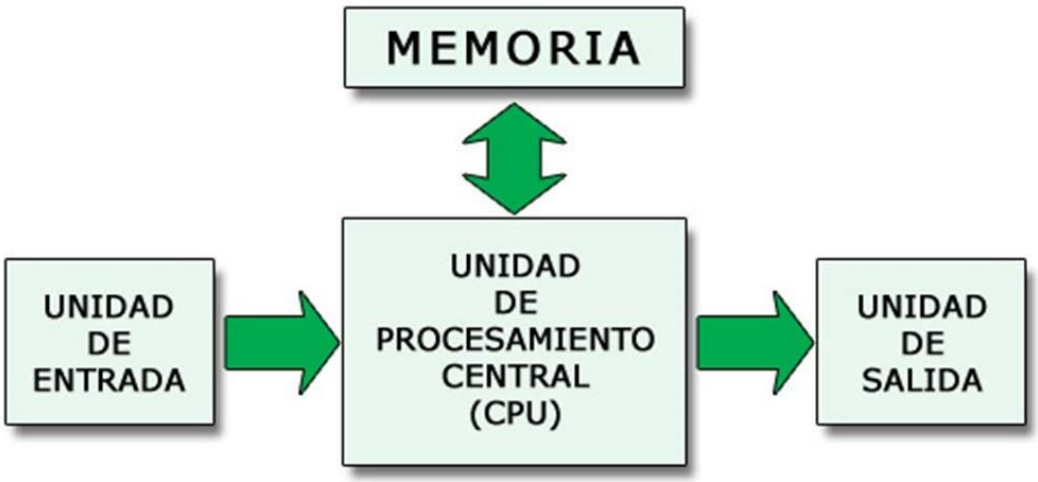
- El hardware, que constituye la parte física del sistema, es decir el ordenador que manejamos con todos sus periféricos.
- El software, que está formado por todo aquello que hace de intermediario entre el usuario y el hardware, a la vez que constituye los trabajos o tareas realizadas en el ordenador. Es decir, esencialmente es cualquier conjunto de órdenes (programas) que se ejecutan en el ordenador destinadas a su funcionamiento (SO.) o a la realización de tareas (aplicaciones).
- Factor humano.

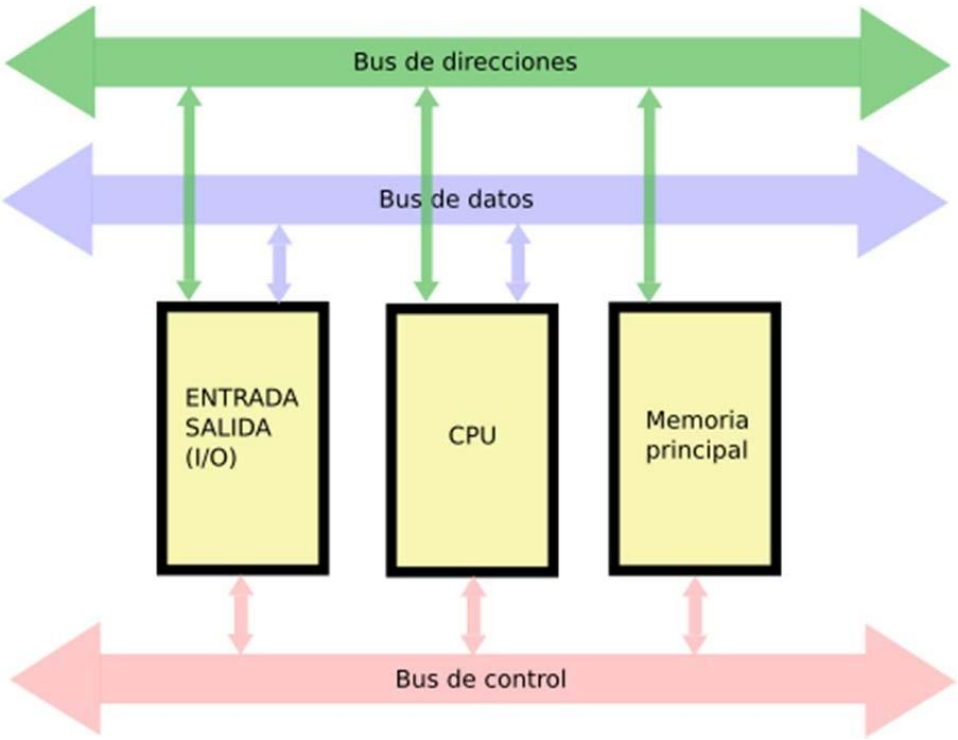
3.1. Elementos funcionales. El Hardware.

El ordenador está compuesto por tres tipos de elementos funcionales: **de operación, de almacenamiento y de interconexión**.

Esencialmente los siguientes:

A) Unidad Central de Proceso o CPU.
B) Memoria Principal.
C) Memoria Secundaria.
D) Dispositivos de E/S.
E) Buses.





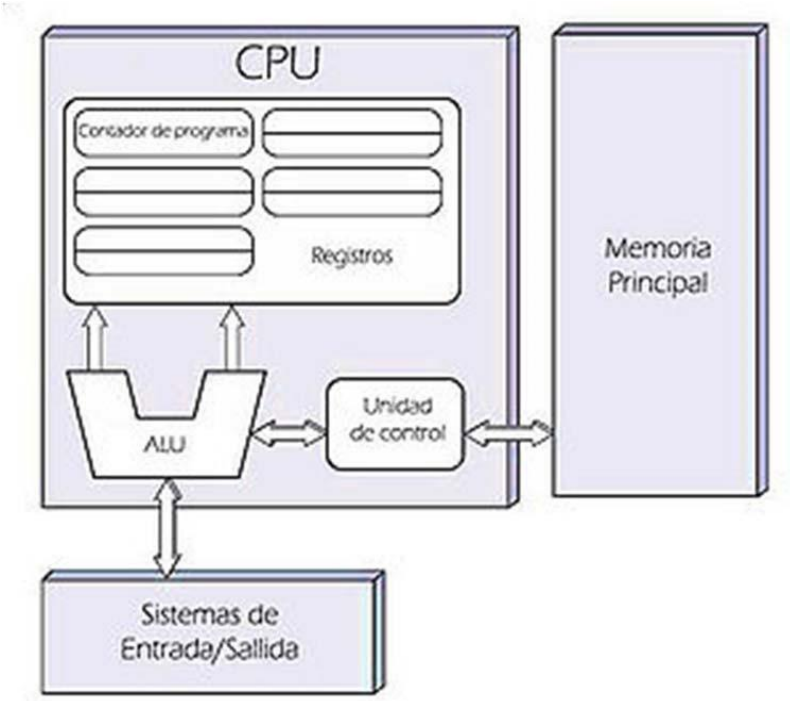
A) Unidad Central de Proceso (CPU):

Es el cerebro del ordenador y sus funciones son:

- Ejecutar las instrucciones que le indique el usuario con el fin de realizar un trabajo determinado.
- Realizar todas las operaciones que sean necesarias para realizar el procesamiento de la información.
- Coordinar, y controlar todos los elementos electrónicos para que no se produzca ningún error.

Dentro de la CPU, se distinguen tres partes bien diferenciadas:

- La *UAL* (Unidad Aritmético Lógica), encargada de realizar las operaciones con los datos.
- La *UC* (Unidad de control), encargada de gestionar el procesamiento de las instrucciones.
- Los registros, o pequeñas unidades de memoria muy rápidas y que tienen un cometido específico cada uno de ellos y que son utilizados internamente por la unidad de control.

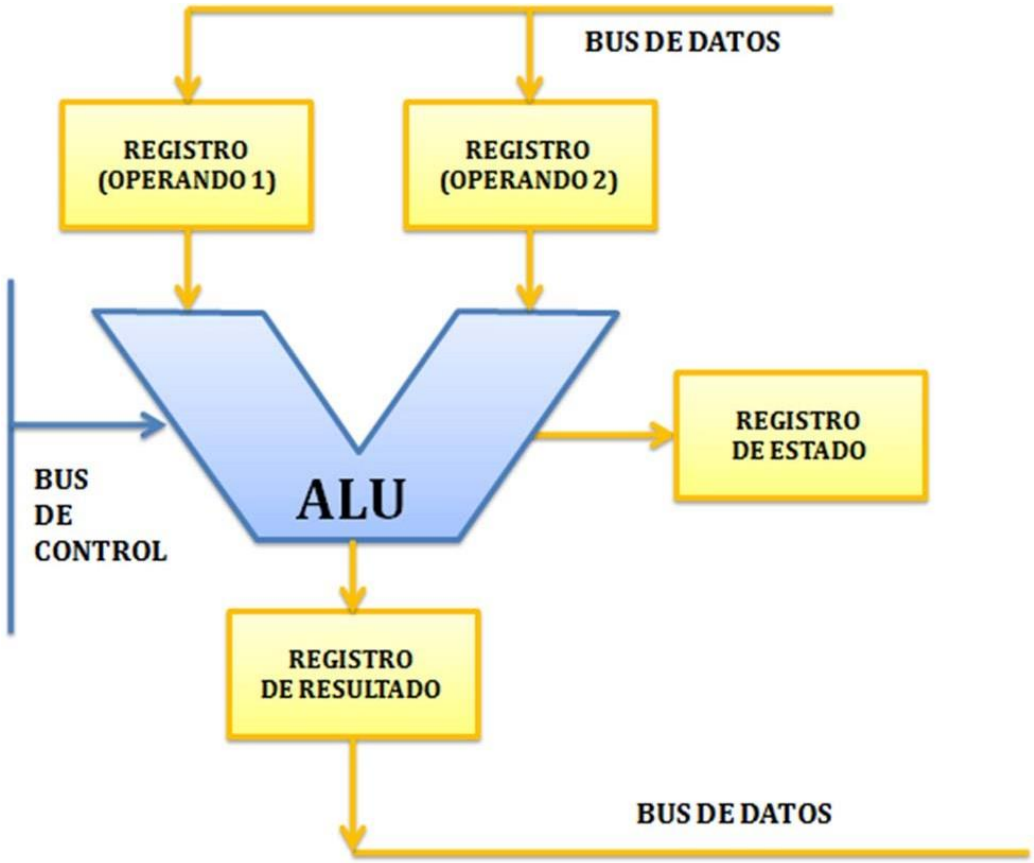


Para desarrollar su trabajo la unidad de control está formada por una serie de elementos. Los más significativos son:

- **Contador de programa** es un registro que guarda la dirección de memoria en la que se encuentra la siguiente instrucción a ejecutar. Al comenzar la ejecución de un programa el contenido del contador de programa es la dirección en memoria de la primera instrucción.
- **Registro de instrucción** contiene la instrucción que se está ejecutando. La instrucción está formada por el código de la operación deseada y por el dato o los datos (o sus direcciones) sobre los que actúa.
- **Reloj** es un circuito especial que proporciona una serie de impulsos a intervalos regulares y que sirve para sincronizar el funcionamiento de los distintos componentes de la Unidad de control para ejecutar una instrucción.

La unidad aritmético-lógica, para desarrollar su trabajo, tiene los siguientes elementos:

- **Registros de entrada**, son registros en los que se almacenan los valores de los operandos afectados por la operación. También se emplean para almacenar resultados intermedios.
- **Circuitos operacionales,** contiene los circuitos electrónicos que permiten la realización de los diferentes cálculos sobre los valores contenidos en los registros de entrada.
- **Registro acumulador,** guarda el resultado de las operaciones efectuadas por el circuito operacional.

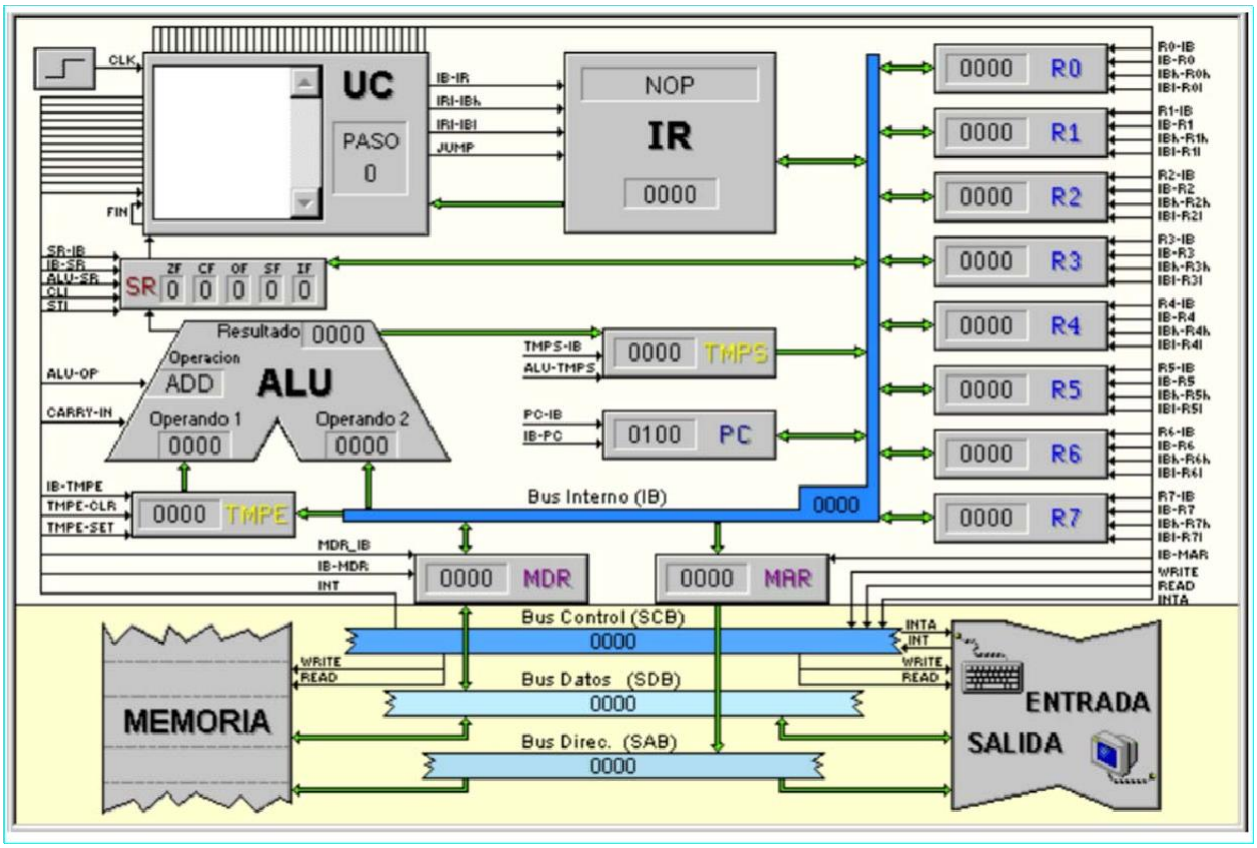


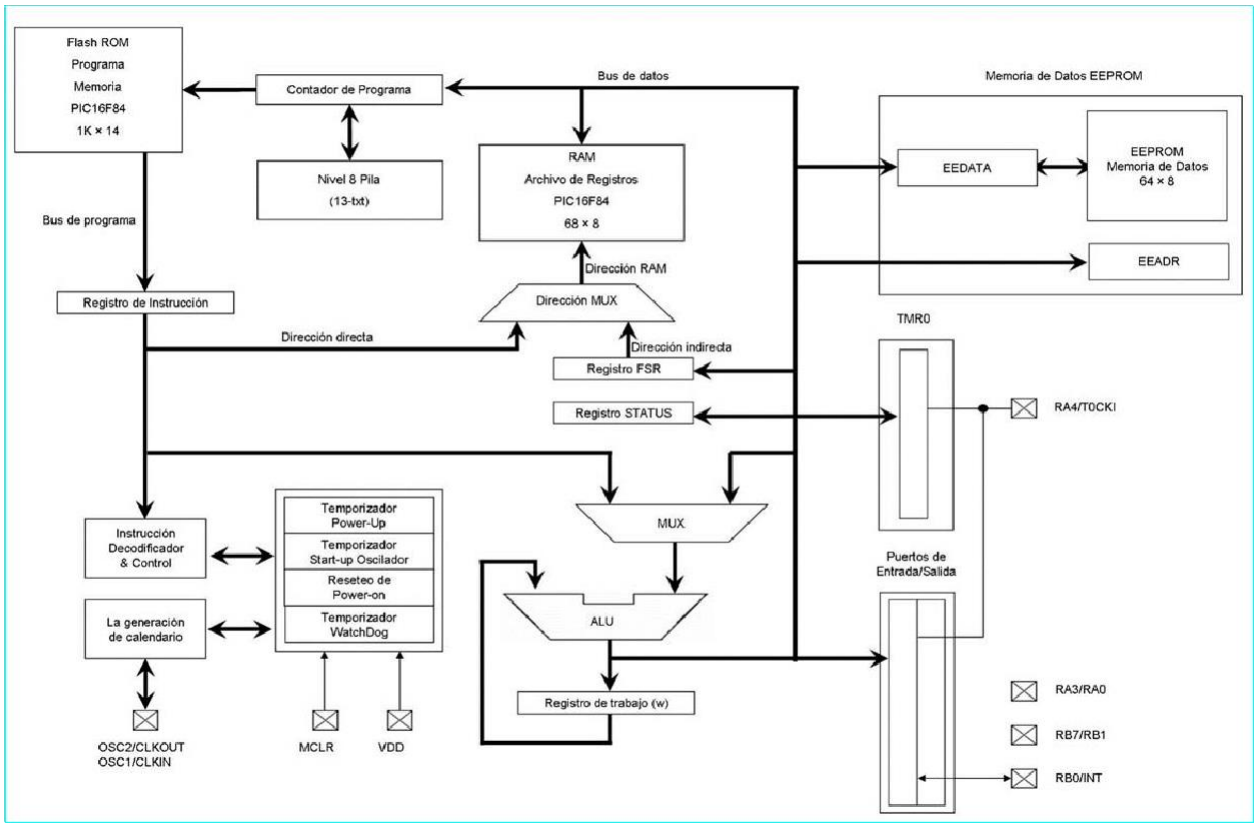
B) Memoria Central o Principal:

Es un almacén de información donde se van a guardar los datos y programas. Está dividida en una serie de registros que se identifican mediante una **dirección**. Todos los registros tienen el mismo número de celdas y en cada una de ellas se almacenará un bit. Al tamaño de cada registro se le conoce con el nombre de longitud del registro (8, 16, 32, 64 bits generalmente)

Para facilitar su uso la memoria tiene dos registros:

- **Registro de dirección de memoria**. Antes de realizar una operación de lectura o de escritura sobre la memoria se ha de colocar en este registro la dirección de la celda sobre la que se va a trabajar.
- **Registro de intercambio de memoria**.
    - Si se trata de una operación de lectura de una información almacenada en la memoria, se copia en este registro el contenido de la posición de memoria señalada por el registro de dirección de memoria.
    - Si la operación es de escritura de un dato en memoria, en este registro se guarda la información que se quiere dejar en la dirección de memoria que guarda el registro de dirección de memoria.


CLK
UC
PASO
0
FIN
IB-IR
IRI-IBh
IRI-IBl
JUMP
NOP
IR
0000
SR-IB
IB-SR
ALU-SR
CLI
STI
SR
ZF CF OF SF IF
0 0 0 0 0
Resultado 0000
Operacion
ALU-OP
ADD
ALU
CARRY-IN
Operando 1
0000
Operando 2
0000
TMPS-IB
ALU-TMPS
0000 TMPS
PC-IB
IB-PC
0100 PC
IB-TMPE
TMPE-CLR
TMPE-SET
0000 TMPE
Bus Interno (IB)
0000
MDR_IB
IB-MDR
INT
0000 MDR
0000 MAR
Bus Control (SCB)
0000
Bus Datos (SDB)
0000
Bus Direc. (SAB)
0000
MEMORIA
WRITE
READ
INTA
INT
ENTRADA
SALIDA
0000 R0
0000 R1
0000 R2
0000 R3
0000 R4
0000 R5
0000 R6
0000 R7
R0-IB
IB-R0
IBh-R0h
IBl-R0l
R1-IB
IB-R1
IBh-R1h
IBl-R1l
R2-IB
IB-R2
IBh-R2h
IBl-R2l
R3-IB
IB-R3
IBh-R3h
IBl-R3l
R4-IB
IB-R4
IBh-R4h
IBl-R4l
R5-IB
IB-R5
IBh-R5h
IBl-R5l
R6-IB
IB-R6
IBh-R6h
IBl-R6l
R7-IB
IB-R7
IBh-R7h
IBl-R7l
IB-MAR
WRITE
READ
INTA


Flash ROM
Programa
Memoria
PIC16F84
1K × 14
Contador de Programa
Nivel 8 Pila
(13-bit)
Bus de datos
RAM
Archivo de Registros
PIC16F84
68 × 8
Memoria de Datos EEPROM
EEDATA
EEPROM
Memoria de Datos
64 × 8
EEADR
Bus de programa
Registro de Instrucción
Dirección RAM
Dirección MUX
Dirección directa
Dirección indirecta
Registro FSR
Registro STATUS
TMR0
RA4/T0CKI
Instrucción
Decodificador
& Control
La generación
de calendario
Temporizador
Power-Up
Temporizador
Start-up Oscilador
Reseteo de
Power-on
Temporizador
WatchDog
MUX
ALU
Registro de trabajo (w)
Puertos de
Entrada/Salida
RA3/RA0
RB7/RB1
RB0/INT
OSC2/CLKOUT
OSC1/CLKIN
MCLR
VDD

C) Memoria Secundaria:

Es un tipo de memoria temporal de muy alta velocidad donde se podrán cargar las aplicaciones y los archivos con los que trabajamos en cada momento.

El que la memoria RAM sea volátil es un inconveniente ya que tendremos que cargar en memoria tanto los programas como los datos con los que vamos a trabajar. Para evitar esto, los ordenadores constan de unos dispositivos de almacenamiento que permiten almacenar de forma permanente tanto los programas como los datos, recuperarlos y llevarlos a la memoria principal cuando queramos ejecutarlos.

Estos dispositivos reciben el nombre de memoria secundaria y la constituyen los disquetes, discos duros, CD-ROM, cintas DAT, etc.

D) Dispositivos o periféricos de E/S

Son los dispositivos que permiten el intercambio de información entre el ordenador y el exterior (normalmente usuarios u otros ordenadores).

Los periféricos se pueden clasificar en:

i. De entrada: Suministran información al ordenador.
   - Teclado.
   - Ratones.

ii. De salida: Informan al usuario sobre los resultados obtenidos en el procesamiento, así como el estado del ordenador.
   - Impresoras.
   - Pantalla.

iii. De entrada/salida:
   - Pantallas táctiles.

Por otro lado, entre los periféricos se distinguen aquellos que permiten el acceso a un sistema de almacenamiento secundario para leer, escribir datos o ambas operaciones a la vez:

- Disqueteras.
- Unidades de disco duro.
- Unidades de CD-ROM.
- Unidades de cinta.

E) Buses:

Son unos circuitos cuyo objetivo es hacer que las instrucciones y los datos circulen entre los distintos dispositivos de un ordenador. Según lleven la información de o desde un elemento a

otro: CPU, memoria y dispositivos de E/S. Podemos distinguir esencialmente tres tipos de buses: de control, de datos y de dirección.

4. El Software.

El software es el conjunto de programas informáticos que actúan sobre el hardware para ejecutar lo que el usuario desee.

Programas: Son un conjunto de instrucciones que hacen que podamos utilizar el hardware del ordenador para almacenar, procesar y recuperar información; encontrar errores de ortografía en un texto; controlar y utilizar los dispositivos hardware del ordenador, etc...

Según su función se distinguen **dos bloques**: sistema operativo, software de programación y aplicaciones.

1. **Sistema Operativo**. Cuyas funciones principales son:
    - Facilitar al usuario el uso de los recursos o hardware del ordenador evitando que este tenga que poseer unos conocimientos profundos sobre cada uno de los dispositivos que forma el ordenador. Convierte el ordenador en máquina virtual.
    - Gestionar adecuadamente los recursos del ordenador para que estos realicen bien el trabajo que se les ha encomendado.
    - Son ejemplos de sistemas operativos: Windows, Linux, Mac OS X ...

2. **El software de programación** es el conjunto de herramientas que nos permiten desarrollar programas informáticos, y **las aplicaciones informáticas** son un conjunto de programas que tienen una finalidad más o menos concreta. Son ejemplos de aplicaciones: un procesador de textos, una hoja de cálculo, el software para reproducir música, un videojuego, etc.

    Programas de aplicación: resuelven los problemas de los usuarios.

    Destacan:

    - **Procesadores de Textos**: Es un tipo de aplicación informática para la creación, edición, modificación y procesamiento de documentos de texto con formato (tal como el tipo y tamaño de la tipografía, adición de gráficos, etc.)

      Ejemplos de procesadores de texto: Write, Word, etc.

    - **Hojas de Cálculo**: Son unos programas que nos permiten realizar operaciones complejas con todo tipo de datos especialmente los numéricos, así como crear gráficos a partir de ellos.

      Ejemplos de hojas de Cálculo: Excel, Lotus123,…

    - **Sistemas Gestores de Bases de Datos**: Son unos programas que nos permiten crear, consultar y modificar una base de datos.

- **Programas de usuario**: Son los que crea el propio usuario para resolver sus problemas particulares, utilizando para ello un lenguaje de programación normalmente de alto nivel (C++, Java, etc.). Cada programa se almacena en un soporte de memoria secundaria mediante una unidad lógica llamada fichero o archivo.

- **Compiladores e Intérpretes**: Son unos programas que traducen un programa escrito en un lenguaje de programación de alto nivel (C++, Java, etc...) a un conjunto de instrucciones máquina que serán las que ejecute el ordenador.

Una aplicación no es otra cosa que un **conjunto de programas**, que están escritos en algún **lenguaje de programación** que **el hardware del equipo** debe interpretar y ejecutar.

Datos de Entrada y de Salida

Los datos de entrada además de ser proporcionados directamente por el usuario al ordenador pueden estar almacenados en los dispositivos de memoria secundaria de donde serán llevados a la memoria principal para su procesamiento por los programas que los vayan a manipular.

Los datos de salida pueden ser presentados directamente al usuario o ser almacenados en los dispositivos de memoria secundaria con el fin de tener la posibilidad de realizar nuevos tratamientos con ellos.

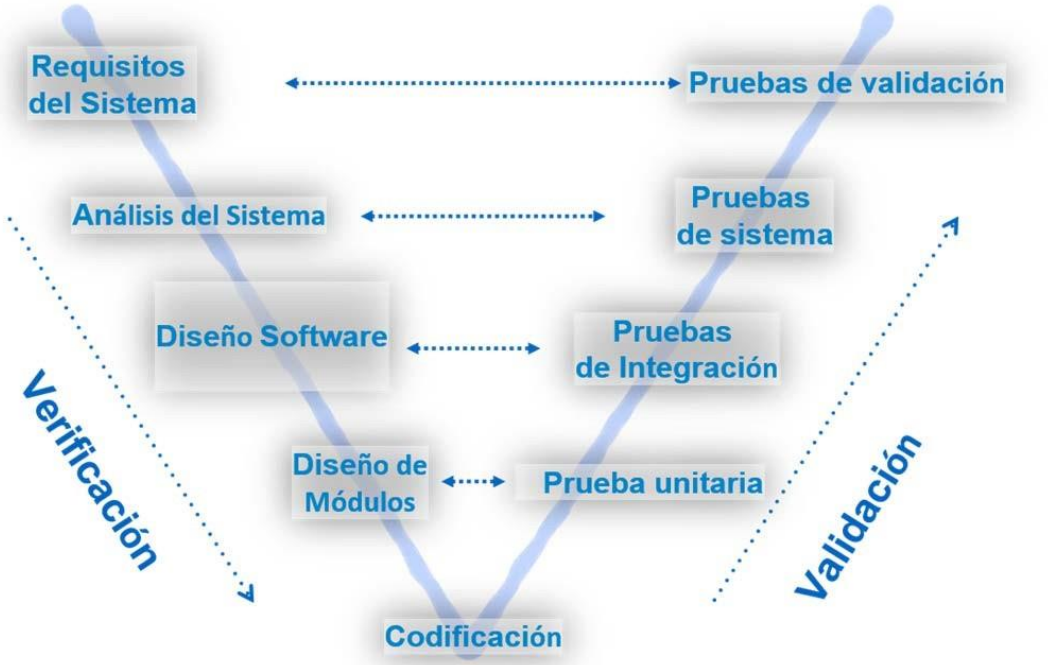
Cuando se decide almacenar los datos de entrada o de salida en algún dispositivo de memoria secundaria, no se hace de forma aleatoria, sino que se almacenan de forma organizada en unas unidades lógicas llamadas **ficheros**. Un fichero o archivo, por tanto, nos va a permitir almacenar en la memoria secundaria información que está relacionada entre sí.

5. ETAPAS EN EL DESARROLLO DE UN PROYECTO.

El ordenador es una máquina que necesita que le digamos lo que tiene que hacer y en qué orden.

Actualmente, los programas se desarrollan en fases o pasos conocidos como **ciclo de vida de una aplicación:**

1. Análisis
2. Diseño
3. Codificación
4. Integración
5. Evaluación y pruebas
6. Explotación
7. Mantenimiento.



El desarrollo de estas fases la hacen equipos de personas denominados: analistas, diseñadores y programadores.

Además, la labor desarrollada por cada uno de estos equipos se refleja en un documento que tendrá que ser contemplado en cada fase posterior y sirve de marco común para el desarrollo del programa:

- Documento requisitos (SRD).
- Documento de diseño (SDD).
- Programa en pseudocódigo.
- Código fuente.
- Documento de cambios.
- Manual de usuario.
- Etc.

En nuestro caso es especialmente importante **la fase de codificación**, ya que es el programador, cumpliendo con los requisitos establecidos y el diseño determinado, quien plantea el conjunto de algoritmos apropiados y los implementa con código, en el lenguaje de programación que corresponde. Posteriormente se integrarán todos los elementos y se realizarán las pruebas correspondientes para verificar la corrección, eficiencia, seguridad, mantenibilidad, etc. del producto final.

La realización correcta de una labor de programación exige, por tanto:

- **Analizar el problema**, es decir, estudiar cuidadosamente el enunciado y características del problema que se quiere resolver programando un ordenador. Por ejemplo, conocer dónde están los datos de entrada, donde deben dejarse los datos de salida, estructura de los datos, límites entre los que varían los datos, establecer con claridad que problema debemos resolver, etc.

- **Diseñar**, o establecer y elegir una estrategia que resuelva el problema planteado. El resultado de esta fase se denomina algoritmo.

- **Codificar el algoritmo**, utilizando un lenguaje de alto nivel. El resultado se denomina programa. Esta etapa se compone de las siguientes fases:

    - **Edición**. En ella y con ayuda del programa editor se escribe en la memoria del ordenador el programa, obteniéndose el programa fuente.

    - **Compilación**. En ella y con ayuda de los programas compiladores o de los programas intérpretes se traduce al lenguaje de la máquina el programa fuente, obteniéndose el programa objeto.

    - **Montaje o linkado**. En ella y cuando en la fase de compilación se utilizó un programa compilador, se añaden al programa objeto las rutinas del sistema y, en algunos casos, los subprogramas que se hayan compilado aparte. El resultado se conoce como programa ejecutable.

    - **Ejecución**. Consistente en probar el correcto funcionamiento del programa con un conjunto de datos de prueba.

    - **Elaboración de la documentación**. Todo programa debe acompañarse con la documentación que permita a los usuarios su utilización, y en la que se indiquen las exigencias de hardware y las limitaciones que tienen los datos.

6. LENGUAJES DE PROGRAMACIÓN.

Un lenguaje de programación es un conjunto de símbolos y reglas para combinarlos que se usan para expresar algoritmos, entendiéndose como algoritmo un procedimiento preciso y no ambiguo que resuelve un problema, y un procedimiento como una secuencia de operaciones bien definidas, cada una de las cuales requiere una cantidad finita de memoria y se realiza en un tiempo finito.

Los lenguajes de programación, al igual que los lenguajes que usamos para comunicarnos, poseen:

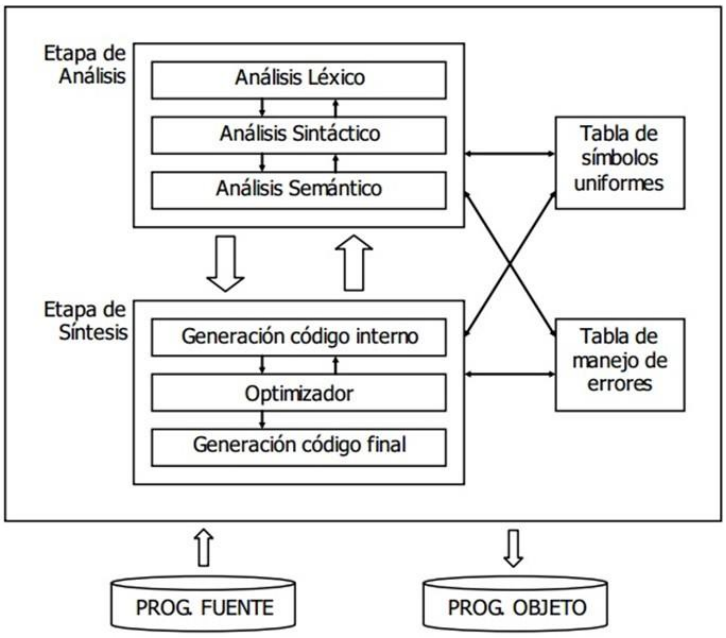
- Un léxico (vocabulario o conjunto de símbolos permitidos).
- Una sintaxis, que indica cómo realizar construcciones válidas del lenguaje.
- Una semántica, que determina el significado de las mismas.

Al desarrollar un programa en un lenguaje de programación distinto del lenguaje que entiende la máquina (binario), es necesario realizar sobre dicho programa una traducción para convertirlo a binario y que así pueda ser entendido y ejecutado por el ordenador. Esta operación de traducción es realizada por unos programas, denominados traductores, que pueden ser de dos tipos: compiladores o intérpretes.

El compilador traduce un programa fuente escrito en un lenguaje de programación, a un programa objeto, escrito en lenguaje máquina.

El programa fuente suele estar contenido en un archivo, y el programa objeto puede almacenarse como archivo en memoria masiva para ser procesado posteriormente, sin necesidad de volver a realizar la traducción.

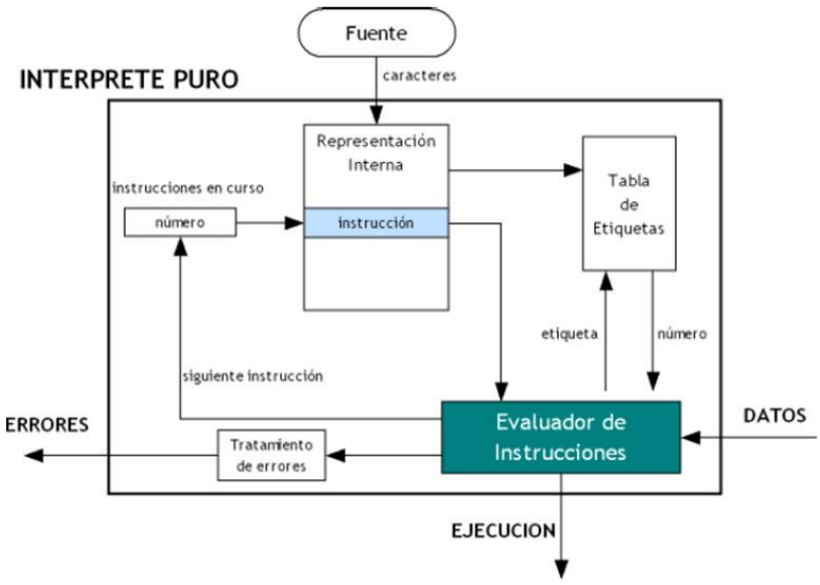
Una vez que el programa ha sido traducido, su ejecución es independiente del compilador, así, por ejemplo, cualquier interacción con el usuario sólo estará controlada por el sistema operativo.



Un intérprete hace que un programa fuente escrito en un lenguaje vaya sentencia a sentencia, traduciéndose y ejecutándose directamente por el ordenador.

El intérprete capta una sentencia fuente, la analiza e interpreta dando lugar a su ejecución inmediata, no creándose por tanto un archivo o programa objeto almacenable en memoria masiva para ulteriores ejecuciones.

Por tanto la ejecución del programa está supervisada por el intérprete.



6.1 Tipos de lenguajes de programación

Los lenguajes de programación se pueden clasificar atendiendo a varios criterios, teniendo en cuenta que puede haber lenguajes que pertenezcan a más de un grupo establecido. Algunos de los criterios a la hora de clasificarlos son los siguientes:

A) Desde el punto de vista cronológico.

Podemos distinguir las siguientes generaciones:

- **Primera generación: Lenguaje máquina.** Corresponde a los años 50 en los que empiezan a aumentar el número de ordenadores y se empieza a abandonar la programación a base de códigos binarios, en lenguaje máquina, para pasar a la programación a base de lenguaje ensamblador mediante el uso de instrucciones asociadas a nemotécnicos.

- **Segunda generación**: **Lenguajes simbólicos y lenguaje ensamblador**. Durante principios de los 60, con el rápido desarrollo del hardware surgen las grandes capacidades de memoria y almacenamiento, a un precio aceptable. Es entonces cuando se empiezan a desarrollar los primeros lenguajes de "alto nivel", que no dependen de la máquina en la que se ejecutan y son más fáciles de usar por el programador, aunque ocupan más memoria y necesitan de más tiempo de ejecución. Podemos citar como ejemplos: Fortran, Cobol, Algol, Basic, etc.

- **Tercera generación: Lenguajes de alto nivel.** A principios de los 70 se pone en práctica la programación estructurada. La programación deja de ser un arte para convertirse en una tarea productiva que se realiza de forma metódica. Ej.: Pascal, Modula-2, C, Ada, Lisp, Prolog, etc.

- **Cuarta generación**: **POO**. Tratan de ofrecer un mayor nivel de abstracción para el programador, prescindiendo por completo del ordenador. Se trata de conseguir que cualquier persona, sin grandes conocimientos de programación, pueda usar estos lenguajes para diseñar pequeñas aplicaciones. Una posible agrupación de estas herramientas de 4ª generación, según el tipo de información que traten, sería: bases de datos, cálculo, etc. Muchos tienen nombres de marcas comerciales. (SQL).

- **Quinta generación: ¿Lenguajes de inteligencia artificial?:** La quinta generación de lenguajes de programación es utilizada para redes neuronales. Una red neuronal es una forma de inteligencia artificial que trata de imitar la mente humana. (Lisp, Prolog, Haskell).

B) Atendiendo a su proximidad al lenguaje de la máquina.

1) Lenguaje máquina o de bajo nivel

El léxico que utilizan estos lenguajes de programación está formado por dos únicos símbolos, el cero y el uno, por lo que la representación de las distintas instrucciones que van a formar parte de un programa, será una combinación de los dos dígitos anteriores. A estas instrucciones se las denomina instrucciones máquina.

Estos lenguajes son dependientes de la arquitectura de la máquina en la que están definidos. Las instrucciones son ejecutadas directamente por la C.P.U. sin necesidad de traducirlas.

Estos lenguajes fueron los primeros que se utilizaron a la hora de programar ordenadores, pero dejaron pronto de utilizarse debido a su complejidad, siendo sustituidos por otros lenguajes más fáciles de aprender y de utilizar, que además reducen la posibilidad de cometer errores.

2) Lenguajes de nivel intermedio o ensambladores

Al igual que el lenguaje máquina, un lenguaje ensamblador es único para cada procesador y sigue estando orientado a la máquina. La gran diferencia entre los dos tipos reside en la representación de las instrucciones. En lugar de las pesadas series de unos y ceros, el lenguaje ensamblador utiliza símbolos reconocibles, llamados nemotécnicos para representar instrucciones.

Ejemplo:

```
ADD   B,1
MOV   A,B
COMP A,E
```

Un programa ensamblador no puede ejecutarse directamente en la máquina, sino que tiene que traducirse al lenguaje máquina, utilizando para ello un programa traductor.

Las ventajas que ofrece un lenguaje ensamblador son las siguientes:

- Los programas escritos en ensamblador aprovechan mejor la memoria y se ejecutan más rápidamente que los lenguajes de alto nivel, que veremos a continuación.
- Se puede trabajar con ellos a nivel de bit, pudiendo acceder a un bit concreto de la memoria, modificándolo si es necesario.
- Se emplean cuando se quiere que una parte o una aplicación entera sea extremadamente rápida.

Entre los principales inconvenientes están:

- Dependencia total de la máquina.
- Programadores muy especializados.
- El programador debe conocer perfectamente el hardware del equipo ya que maneja directamente las posiciones de memoria, registros del procesador y demás elementos físicos.

3) Lenguajes de alto nivel

Son los lenguajes que nos permiten describir algoritmos mediante programas para ser ejecutados en el mayor número de ordenadores, no siendo necesario conocer el funcionamiento interno de la máquina.

Los lenguajes de alto nivel poseen una potencia considerable en sus instrucciones, de modo que cada instrucción en un lenguaje de alto nivel equivale a varias en lenguaje máquina.

Estas instrucciones normalmente están escritas en inglés y no en códigos nemotécnicos, y son independientes de la máquina con la que se opera.

Las instrucciones de los lenguajes de alto nivel están orientadas a la resolución del problema y no a la máquina que lo va a resolver, por lo que es necesario traducirlas a código máquina.

Ejemplo:

```
#include <stdio.h>
void main(void)
{
        printf("Este programa está hecho en C”);
}
```

Con estos lenguajes empiezan a surgir los entornos de programación, que integran en una sola herramienta editores para crearlos, traductores y depuradores, incluso en máquinas distintas de la de destino.

Ofrecen las siguientes ventajas:

- El tiempo de formación de los programadores es relativamente corto en comparación con el necesario en los lenguajes de bajo nivel.
- El tiempo para codificar y poner a punto un programa en lenguaje de alto nivel es inferior al empleado con los lenguajes de bajo nivel.
- Los cambios y correcciones resultan más fáciles.
- Se reduce el coste de creación y mantenimiento de los programas.

Inconvenientes:

- Se incrementa el tiempo de ejecución, pues hay que traducir los programas al lenguaje máquina.
- No se aprovechan las posibles ventajas de la arquitectura interna del sistema.
- Se incrementa la ocupación de memoria interna pues además del programa escrito en lenguaje de alto nivel, necesitamos en memoria el traductor del lenguaje al lenguaje máquina.

C) Según la filosofía con lo que fueron concebidos.

1) Lenguajes Estructurados

Utilizan los principios de la programación estructurada (Un único principio y un único fin, y toda instrucción se representa utilizando estructuras secuenciales, condicionales y repetitivas).

2) Lenguajes Modulares

Utilizan el diseño top-down, descendente, de arriba abajo, dividiendo el programa en módulos más elementales (Divide and Conquer) para el desarrollo de programas.

3) Lenguajes Orientados a Objetos

El diseño de los programas se basa más en los datos que en los procesos, la unidad de proceso es el objeto y en él se incluyen los datos y las operaciones que actúan sobre ellos. Además de las técnicas de programación estructurada, y modular, incorpora otras como la herencia, encapsulación y el polimorfismo.

4) Lenguajes de Marcas (HTML)

El HTML es el lenguaje utilizado para crear y reconocer documentos hipermedia a los que accedemos a través de la red Internet (páginas web). Estos documentos poseen la extensión "html".

Este lenguaje consiste en un sistema de introducción de marcas o delimitadores de referencia en un fichero de texto, utilizado para la creación de documentos hipertexto.

5) Lenguajes orientados a la Arquitectura Cliente-Servidor

El auge de Internet ha potenciado la aparición de aplicaciones Cliente/Servidor, que se ejecutan en servidores web. Un ejemplo de este tipo de aplicaciones, son las aplicaciones CGI (Common Gateway Interface) que tienen como objetivo poner en contacto a un servidor web con un ordenador cliente para proporcionarle los datos obtenidos de una búsqueda solicitada previamente por el cliente o para pedirle, mediante un formulario, una serie de datos con el objetivo de actualizar una base de datos en el servidor.

Una característica común a todas estas aplicaciones es que en el ordenador cliente las peticiones que se hacen al servidor y los resultados obtenidos en las mismas se visualizan como parte de una página web (htm), lo que hace que sean multiplataforma.

Entre los principales lenguajes que se utilizan para la construcción de este tipo de aplicaciones están:

- Java: Lenguaje desarrollado por Sun Microsystems para la elaboración de aplicaciones exportables a la red y capaces de operar sobre cualquier plataforma a través, normalmente, de visualizadores o navegadores de páginas WWW. El programa Java se descarga desde el servidor Web y lo interpreta un programa que se ejecuta en el equipo que contiene el explorador de Web.

- Perl: Lenguaje de programación muy utilizado para la elaboración de aplicaciones CGI. Es multiplataforma y funciona bajo UNIX. Perl es un lenguaje de programación muy sencillo, desarrollado inicialmente por Larry Wall.

- ASP: (Active Server Pages). Es un lenguaje independiente, diseñado por Microsoft para la codificación eficiente de los scripts de los servidores, que fueron diseñados para ser ejecutados por un servidor Web en respuesta a la petición de un URL de un usuario. Los scripts de ASP son similares a otros scripts de servidores con los que puedes estar familiarizado, que son utilizados en otras plataformas, como Perl, Python, etc.

  Microsoft también utiliza Visual Basic Script para la creación de este tipo de aplicaciones.

- PHP: Acrónimo recursivo en inglés de PHP Hypertext Preprocessor (preprocesador de hipertexto), es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en un documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera el HTML resultante.

D) Según el interfaz de usuario en el que se van a ejecutar los programas.

1) Lenguajes para interfaz de usuarios orientados a caracteres (CUI)

Los programas desarrollados con estos lenguajes sólo se preocupan de ofrecer un alto grado de computabilidad, es decir, de realizar la función para la que fue construido de la forma más eficiente y eficaz posible, sin preocuparse de facilitar a los usuarios el manejo de los mismos. De hecho la interfaz de estos programas con los usuarios es casi inexistente, aunque en algunos casos se trata de un menú de opciones. La comunicación entre el usuario y la aplicación se restringe únicamente a la utilización del ratón y del teclado. Dentro de este grupo se pueden incluir lenguajes como: BASIC, PASCAL, C.

2) Lenguajes para interfaz Gráficos de usuarios (GUI)

En los programas desarrollados con estos lenguajes es tan importante su capacidad para computar como su capacidad para comunicarse con el usuario. Ahora éstos interactuarán con las aplicaciones no sólo a través del teclado y la pantalla, sino que podrán utilizar nuevos elementos como el ratón, menús botones, ventanas, etc., que harán que dicha interacción sea más fácil y cómoda.

3) Lenguajes para interfaces multimedia o lenguajes de autor

Las interfaces multimedia son las que se utilizan actualmente y se pueden considerar GUI las que se han ampliado con elementos multimedia (sonido, vídeo, animaciones, etc.). Entre los lenguajes de autor más utilizados actualmente para el desarrollo de aplicaciones multimedia están:

a) Director de Macromedia: Las aplicaciones construidas con este lenguaje de autor, se desarrollan como si fuesen películas, de tal forma que cada pantalla con los elementos multimedia que la forman se identifica con un frame o fotograma.

b) ToolBook: En este caso la aplicación se ve como si fuese un libro, y las distintas pantallas de la aplicación como páginas.

7. PARÁMETROS PARA MEDIR LA CALIDAD DE LOS PROGRAMAS

Cuando se desarrolla un programa en ocasiones es difícil determinar si el trabajo realizado es el correcto o si se va por el buen camino en las distintas versiones que se van realizando. Si bien ya se ha visto en esta unidad las distintas etapas por las que pasa un proyecto, en determinados momentos habrá que pararse a pensar si lo desarrollado es lo correcto, concretamente en las etapas de análisis.

Para poder medir la calidad se han de estudiar una serie de factores, los cuales no siempre se ven con claridad o no son medibles, pero que pueden incluso ayudarnos para orientar el proyecto hacia mejoras evolutivas considerables. Estos factores son:

- Concreción: grado de cumplimiento con las especificaciones.
- Fiabilidad: grado de ausencia de fallos (error, warnig, dessaster,…).
- Eficiencia: relación entre el nº de resultados y cantidad recursos usados.
- Seguridad: dificultad de acceder a la información por terceros no autorizados.
- Facilidad de uso: nivel de esfuerzo para aprender el manejo.
- Mantenibilidad: facilidad para corregir fallos del producto.
- Flexibilidad: facilidad para modificar el producto.
- Facilidad de prueba: mide el esfuerzo realizado para probar el producto.
- Portabilidad: medida en que el producto puede variar de plataforma.
- Reusabilidad: facilidad de usar partes del producto en otros proyectos.
- Interoperatividad: facilidad para ser compatible con otros productos.
- Accesibilidad: posibilidad de acceder al producto.
- Usabilidad: si realmente es útil o no.