

## INTRODUCCIÓN

### **1-. Tipos de Información**

Las magnitudes o variables analógicas tienen un rango de variación continuo de valores, pudiendo tomar cualquier valor dentro de un intervalo definido en el campo de los números reales. Las magnitudes o variables digitales sólo pueden tomar valores discretos, estando sus valores definidos dentro de intervalos fijados en el campo de los números enteros, pueden representarse exactamente mediante un número finito de cifras.

El concepto de señal implica la dependencia entre dos variables o magnitudes. Las señales más frecuentes son las que tienen al tiempo como variables independiente y se les denomina señales dependientes del tiempo. Estas se dividen en continuas o discretas según la forma en que evoluciona la variable independiente del tiempo. Las señales continuas se denominan analógicas y se caracterizan por que su valor puede cambiar en cualquier instante, las señales discretas en el tiempo (digitales) su valor sólo puede cambiar en instantes específicos (la señal se define en estos instantes).

Para que los computadores adquieran la capacidad de procesar la información proveniente de la observación de fenómenos naturales se deben de convertir las señales continuas en discretas. Proceso que se denomina muestreo, conversión analógico-digital o AD.

La conversión de una señal discreta en continua, se utiliza cuando se pretende que el computador comande o controle sistemas físicos o máquinas. Este proceso se denomina bloqueo, conversión digital-analógica o DA.

Las señales discretas suelen depender de un número entero  $n$ , por lo que se debe definir un periodo de conversión  $T$ , para convertir los números en tiempos y poder generar señales continuas. Según la forma de la función de conversión  $f$  el bloqueador será causal o no causal. Bloqueador causal es aquél que utiliza para realizar la construcción de la señal continua valores de la señal discreta anteriores a ese instante. Bloqueador no causal es aquél que utiliza para realizar la construcción además de los puntos anteriores a dicho instante, algún valor de la señal discreta posterior a dicho instante. Los bloqueadores causales se utilizan en procesos en los que las muestras de la señal discreta van apareciendo sucesivamente en el tiempo, y por tanto, los valores posteriores al instante de la construcción están disponibles. Los bloqueadores no causales se utilizan cuando se dispone de un registro completo de la señal discreta y se presentan ventajas en los procesos de reconstrucción de señal.

### **2-. Computadores analógicos.**

Procesan información analógica, es decir, utilizan magnitudes y señales analógicas. Están basados en un componente electrónico denominado amplificador operacional.

Los bloques de un computador analógico son los circuitos electrónicos de cálculo que realizan operaciones. Estos computadores se programan por hardware.

El funcionamiento de los elementos de cálculo está gobernado por una unidad de control que envía señales de arranque y parada según los pulsadores del panel de control de la máquina. En términos generales los computadores analógicos pretenden simular el comportamiento del mundo físico (descrito por ecuaciones diferenciales).

### **3-. Computadores digitales**

Procesan información digital, sus magnitudes y señales son digitales. Los computadores digitales son apropiados para simular el comportamiento humano.

Sus elementos principales son la Unidad de Memoria (UM), Unidad de Control de programa (UC) y la Unidad Aritmético-Lógica. Estas dos últimas se unen formando la Unidad Central de Proceso. La comunicación con el exterior la realiza por la unidad de entrada salida. Cualquier computador digital debe de disponer de los siguientes elementos:

1. Un procesador capaz de interpretar y ejecutar programas.
2. Una memoria para almacenar programas y datos.
3. Un sistema de transferencia de información entre el computador y el exterior.

### **4-. Desarrollo histórico de máquinas analógicas.**

Se empezó utilizando elementos pasivos, que presentaban el inconveniente de no poder amplificar la potencia de las señales. La limitación de velocidad por el uso de componentes electromecánicos se resolvió con la aparición del amplificador de corriente continua, llamado amplificador operacional (1.940). Constituye la base de los computadores analógicos universales utilizados en operaciones de cálculo y simulación, y también es el elemento fundamental de los dispositivos electrónicos de regulación automáticos y filtrado de señales.

## REPRESENTACION DE LA INFORMACION (I)

### 1-. Introducción.

**Concepto:** Los computadores digitales actuales se basan en una tecnología electrónica que permite representar los datos mediante combinaciones de circuitos del tipo flip-flop o biestables. Estos elementos básicos sólo admiten dos estados. Cada uno de los elementos de la información binaria recibe el nombre de bit.

La codificación consiste en establecer unas reglas que definen una correspondencia, que da lugar a tipos diferentes de códigos. Dichos criterios se denominan sistemas de codificación.

**Clasificación de la información:** Existen dos flujos de información en un computador digital. Uno es el flujo de datos que han de ser manipulados para producir los resultados, y otro es el flujo de instrucciones de máquina, denominado flujo de control, cuyo objetivo es expresar las manipulaciones a realizar en dicho datos.

Los datos pueden ser de naturaleza diversa numéricos o alfabéticos. El flujo de control se compone de las ordenes que debe ejecutar el computador y, que recibe el nombre de códigos de instrucción. La naturaleza de la información influye sobre el sistema de codificación elegido para su representación.

### 2-. Sistemas de codificación.

**Directa:** Consiste en establecer una correspondencia biunívoca entre un conjunto de símbolos y un conjunto de códigos binarios mediante una tabla.

**Por campos:** Se descompone la secuencia de cada código en campos. Es más sencilla que la directa.

**Secuencia de códigos. Códigos de control:** Cuando los elementos de información no se procesan o almacenan sino en conjunto, los códigos de los sucesivos datos se tratan o registran secuencialmente. Con esto se abre la posibilidad de que determinados códigos especiales modifiquen la interpretación de los que aparecen a continuación. La condición de que los símbolos que se representan mediante los códigos de control deban aparecer con poca frecuencia es un criterio fundamental para decidir la codificación en este sistema.

### 3-. Códigos numéricos.

#### Fundamentos de los sistemas de numeración.

Un elemento fundamental que caracteriza a todos los sistemas de numeración es la base, que es el número de símbolos que se utilizan para realizar la representación. Se denomina rango de representación al conjunto de cantidades representables.

#### Sistemas posicionales.

El significado de los símbolos varía en función de la posición que ocupen en la cadena.

#### Representación de números negativos.

**Módulo y signo:** El bit de signo se coloca antes de la cifra de mayor orden y se le asigna el valor 0 si el número es positivo y 1 si es negativo. Características.

- La multiplicación y la división se tratan sin dificultad.
- Posibilidad de desbordamiento de la suma y la resta.
- Rango asimétrico  $[-2^{n-1}+1, 2^{n-1}]$ . El cero tiene doble representación.
- Extensión de signo complicada
- Complicación en la suma y la resta.

**Complemento a 1:** Utiliza el bit de la izquierda para el signo. El positivo es igual que en módulo y signo. Los números negativos se obtienen complementando todos sus dígitos. Características:

- El cambio de signo se reduce al complemento lógico.
- La suma es sencilla. Se debe tener en cuenta que si aparece un acarreo en la posición  $n$  se debe incrementar en una unidad el resultado.

$$\begin{array}{r}
 111001010 \quad -53 \\
 111101000 \quad -23 \\
 \hline
 1110110010 \quad -76 \\
 \downarrow \text{acarreo} \\
 1 \\
 \hline
 110110011
 \end{array}$$

**Complemento a 2:** Se realiza el complemento a 1 y se le suma 1. Características:

- El cambio de signo es sencillo
- Las sumas y restas son sencillas.
- Se debe controlar la posibilidad de desbordamiento en la suma y la resta. Aunque es fácil de detectar, ya que sólo puede producirse en sumas con números del mismo signo (o restas del contrario), y el bit de signo del resultado indique un signo distinto al esperado:

111001011	-53	000110101	53
<u>111101001</u>	<u>-23</u>	<u>111101001</u>	<u>-23</u>
1110110100	-76	1000011110	30
└─→	Acarreo	└─→	Se desprecia

- Complicación de la multiplicación y la división al tener que considerar los operandos complementados.
- Rango asimétrico  $[-2^{n-1}, 2^{n-1}-1]$ . El cero tiene una única representación.
- La extensión de signo se limita a repetir el bit de la izquierda.

*Exceso a M:* Se incrementan los números en M y el resultado se da en binario puro.

### Representación de números reales:

*Coma fija:* La posición está fijada de antemano y es invariante.

*Coma flotante:* La posición de la coma es variable dependiendo del valor del exponente. Es de la forma:

$m \cdot 10^{\text{exp}}$  (En decimal)       $m \cdot 2^{\text{exp}}$  (En binario)

Tiene dos campos uno contiene el valor de la mantisa y el otro de valor del exponente. El bit más significativo de la mantisa contiene el signo. Existen tres formatos:

Signo	Mantisa	Exponente	→ Directo
Signo	Exponente	Mantisa	→ Comparación rápida
Signo	Exponente	Signo      Mantisa	→ Precisión ampliada

Con el fin de salvar la ambigüedad de la representación en coma flotante, ya que un valor puede tener más de una representación, se normaliza la representación haciendo que el primer bit significativo de la mantisa ocupe la posición inmediatamente a continuación del signo.

Trabajando sólo con mantisas normalizadas se cumple siempre que el primer bit de la mantisa es siempre el complemento del bit de signo, por lo que no es necesario incluirlo en la codificación, pudiendo ampliarse la precisión al disponer de un bit adicional en la mantisa. El bit que no se incluye recibe el nombre de bit implícito. Las características de los sistemas de representación en coma flotante son:

- El exponente se representa en exceso a  $2^{n-1}$ , siendo n el número de bits del exponente.
- La mantisa es un número real normalizado, sin parte entera. Su representación puede ser en cualquier sistema: módulo y signo, Complemento a 1 o Complemento a 2.
- La base de exponenciación es una potencia de dos.

*Representación en simple precisión: Palabra de 32 bits.*

Signo	Exponente	Mantisa
31	30    23	22    0
1 bit	8 bits	23 bits

*Representación en doble precisión: Palabra de 64 bits.*

Signo	Exponente	Mantisa
64	63    52	51    0
1 bit	11 bits	52 bits

*Precisión en la información:* Resolución o distancia entre número consecutivos: en los sistemas de coma fija la resolución es uniforme en todo el rango e igual a la cifra menos significativa del código:  $2^{-f}$  siendo f el número de cifras fraccionarias.

En los sistemas de coma flotante varía a lo largo del rango, en virtud del valor del exponente. Considerando sólo la mantisa la resolución es uniforme y de valor  $2^{-m}$ , si consideramos el exponente será de  $2^{e-m}$ , donde e es el valor decimal del exponente.

Precisión: Error en la representación: si un número se aproxima por su representación más cercana, el error absoluto  $E_a$  será igual o menor que la mitad de la resolución.

En coma fija       $E_a = |x - x'| = 2^{-f-1}$

En coma flotante       $E_a = |x - x'| = 2^{e-m-1}$

Error relativo: Es el cociente entre el error absoluto y el valor a representar:

$$E_r = \frac{|x - x'|}{x} = \frac{2^{e-m-1}}{2^{e-1}}$$

### IEEE 754

- Mantisa fraccionaria normalizada
- Mantisa en módulo y signo
- Mantisa en precisión ampliada, el bit implícito siempre vale 1
- El exponente en exceso a  $2^{n-1}-1$

## REPRESENTACION DE LA INFORMACION II

### 1-. Códigos alfanuméricos.

#### *Características:*

- Juego de caracteres:
  - Letras del alfabeto (Mayúsculas y minúsculas )
  - Las 10 cifras del sistema decimal.
  - Signos de puntuación
  - Caracteres de control
- Longitud del código entre 6 y 12 bits
- Sistema de codificación directa
- Número máximo de caracteres distintos es de  $2^{\text{longitud}}$ .

#### *Códigos de 6 bits:*

- 64 caracteres:
  - 26 Letras
  - 10 cifras
  - 28 caracteres especiales

El primer sistema de este tipo fue el código BCD alfanumérico. Fue una extensión del BDC numérico, con dos bits por la izquierda.

#### *Códigos de 7 bits:*

El más representativo es el código ASCII. Los siete bits dan lugar a 128 combinaciones diferentes:

64 Letras mayúsculas y cifras  
32 códigos de transmisión  
32 letras minúsculas.

#### *Códigos de 8 bits:*

El más representativo es el EBDIC, que se utiliza como código interno, tiene 256 caracteres aunque sólo se utiliza la mitad.

### 2-. Datos etiquetados

En la mayoría de computadores, los tipos y las estructuras de los datos son implícitos, es decir, los datos se representan a título individual, sin ninguna información sobre su tipo ni sus interrelaciones siendo los programas que los emplean los que deben definir el tipo de los datos almacenados y las relaciones entre ellos.

Otro sistema es dotar a cada palabra de memoria del computador de una etiqueta (tag) que indica el tipo de contenido. La ventaja de este sistema es la simplificación de las instrucciones del programa, ya que no tienen que saber de que tipo de dato se trata. Pero encarece el computador.

Estas máquinas etiquetadas se adaptan mucho mejor que los computadores convencionales a los lenguajes de programación que comprueban el tipo de operandos de forma dinámica ( tiempo de ejecución).

### 3-. Códigos de instrucciones:

Los códigos numéricos y alfanuméricos permiten representar dentro del computador la información que se debe procesar (datos). Las instrucciones se codifican por campos. Normalmente hay un primer campo, de longitud fija, que codifica el tipo de operación que se debe efectuar, seguido de los campos necesarios para especificar qué operandos hay que utilizar y donde hay que guardarse el resultado de la operación.

En un computador coexisten diferentes formatos de códigos de instrucción ya que el tipo de operación puede llevar implícita la situación de uno o varios operandos y frecuentemente la del resultado.

### 4-. Representación redundante. Corrección de errores.

Una codificación en la que todas o casi todas las combinaciones binarias posibles tengan significado se denominan código denso, y no admite detección de errores pues cualquier alteración en una combinación de bits conducirá a otra combinación válida que será aceptada, siendo erróneo el resultado.

Por el contrario un código en el que sólo ciertas combinaciones sean significativas permitirá la posibilidad de detección diciéndose que el código es redundante. La redundancia por si sola no garantiza la detección del error.

Se define como distancia entre dos combinaciones binarias al número de bits que deben ser modificados en una de las combinaciones para obtener otras. La característica que establece el grado de redundancia de un código es la distancia de un código binario, que se define como la menor de las distancias entre dos combinaciones binarias cualesquiera del mismo. Para que un código sea capaz de detectar algún tipo de error, su distancia debe ser superior a la unidad.

La distancia del código está íntimamente ligada al número de errores que se pueden detectar en el mismo: Si la distancia es  $n$  se podrán detectar errores simultáneos de  $n-1$  bits como mucho. Los códigos detectores, además de detectar el error, lo corrigen. Sin embargo necesitan más bits que los códigos detectores. La distancia mínima que debe tener un código para poder corregir errores de  $n$  bits es al menos de  $2n + 1$  bits.

La paridad de un código binario está relacionada con el número de unos que tiene éste. Si el número de unos es par, el código es de paridad par. Si el número de unos es impar, el código será de paridad impar. Si se añade un bit de paridad a un código denso, la distancia del código pasa a ser de 2 y se detectan errores de un bit. El bit adicional de paridad se pone a cero o uno de forma que el número de unos sea siempre par ( o impar). El sistema de utilizar un bit de paridad en cada código recibe el nombre de paridad transversal. Cuando se transmiten secuencias de códigos, se simplifica la detección de errores mediante la adición de un código más al final de la secuencia. Este código contendrá los bits de paridad correspondientes a las columnas de bits del mismo orden de todos los códigos de la secuencia. Es lo que se conoce como paridad longitudinal o de bloque. La generación de esta secuencia adicional, y la posterior detección de errores se realiza fácilmente mediante el empleo de un operador o-exclusivo o la operación suma.

### Códigos de Hamming:

Son códigos correctores de errores, cuya distancia mínima es 3, que permiten detectar errores de 2 bits y corregir errores de 1 bit. Supongamos que queremos enviar un dato de 6 bits, 010010. Elegimos por convenio transmisión con paridad PAR. ( Con paridad IMPAR es análogo ).

- El primer paso será determinar el número de bits de paridad que necesitaremos adjuntar para la detección del posible error y su corrección. Este número vendrá dado al resolver la inecuación:

$$2^p \geq n + p + 1$$

donde  $n$  es el número de bits del que se compone el dato y  $p$  es el número de bits de paridad necesarios. En nuestro caso,  $p$  tendría el valor 4, así que transmitiríamos 10 bits. 6 de dato, más los 4 de paridad.

- El segundo paso sería determinar la posición que ocupará cada bit. Los bits de paridad ocuparán siempre una posición determinada, y que son las potencias de 2. Así como necesitamos 4 bits de paridad, ocuparían las posiciones  $2^0, 2^1, 2^2, 2^3$  que son respectivamente 1, 2, 4 y 8.

Con todo esto ya tendríamos la estructura de lo que deberíamos transmitir:

Bit	1	2	3	4	5	6	7	8	9	10
Bits de paridad	P1	P2		P3				P4		
Bits de datos			D1		D2	D3	D4		D5	D6
Nuestro ejemplo			0		1	0	0		1	0

- El tercer paso es determinar los bits de paridad. Para ello necesitamos montar una tabla como la siguiente:

Orden	Binario	Bit	A	B	C	D
1	0001	P1	*			
2	0010	P2		*		
3	0011	D1	*	*		
4	0100	P3			*	
5	0101	D2	*		*	
6	0110	D3		*	*	
7	0111	D4	*	*	*	
8	1000	P4				*
9	1001	D5	*			*
10	1010	D6		*		*

Esto significará que el primer bit de paridad P1 controla los bits de datos D1, D2, D4 y D5, que son los que hemos marcado con él. El bit de paridad P2 controla los bits de datos D1, D3, D4 y D6. De esta manera podremos calcular el valor de cada bit de paridad. En el caso de P1, como controla a D1, D2, D4 y D5 que valen respectivamente 0, 1, 0 y 1, el bit de paridad vale 0. Análogamente calculamos los otros obteniendo:

		D1	D2	D3	D4	D5	D6	
P1	Controla	0	1		0	1		P1=0
P2	Controla	0		0	0		0	P2=0
P3	Controla		1	0	0			P3=1

P4	Controla					1	0	P4=1
----	----------	--	--	--	--	---	---	------

Con esto ya tendríamos el resultado de nuestra transmisión, que tendría que ser:

Bit	1	2	3	4	5	6	7	8	9	10
Bits de paridad	P1	P2		P3				P4		
Bits de datos			D1		D2	D3	D4		D5	D6
Nuestro ejemplo	0	0	0	1	1	0	0	1	1	0

Imaginemos que en lugar de recibir 0001100110 hubiésemos recibido un error en el 7º bit, o sea:0001101110.

¿ Cómo hubiésemos detectado el error y hubiésemos podido corregir ? Montaríamos una tabla como la siguiente:

Bit recibido	Orden en binario	
0	0001	- - - 0
0	0010	- - 0 -
0	0011	- - 0 0
1	0100	- 1 - -
1	0101	- 1 - 1
0	0110	- 0 0 -
1	0111	- 1 1 1
1	1000	1 - - -
1	1001	1 - - 1
0	1010	0 - 0 -
		0 1 1 1

Situamos el valor del bit recibido en las posiciones que hay un 1 en el orden en binario, y nada donde existe un 0 en el orden binario. O sea, para el primer bit recibido (0), como el orden en binario es 0001, pondríamos - - - 0.

Entonces calcularíamos el bit de paridad de las cuatro columnas obtenidas, dando 0 1 1 1 que en binario equivale al número 7 decimal, precisamente el bit que había erróneo. Complementando dicho bit, tendríamos el resultado correcto.

En general, cuando se recibe el paquete compuesto por bits de datos bits de paridad, debemos comprobar la paridad del conjunto.

En el supuesto de haber elegido paridad PAR, tendríamos que:

- Se recibe paridad IMPAR ( nuestro ejemplo )→ El valor obtenido 0111 en el 'checksum' es donde se encuentra el error.
- Si se recibe paridad PAR, caben 2 posibilidades. 'Checksum' = 0 → No hay error  
'Checksum' ≠ 0 → 2 Errores. No se puede corregir.

**Códigos mayoritarios:** Consiste en repetir la información un cierto número de veces, si existen discrepancias, el código se decidirá por mayoría. El número de repeticiones es normalmente impar para evitar empates.

### 5-. Compactación de la información.

**Codificación diferencial:** Se utiliza cuando las unidades de información sucesivas difieren poco entre sí. En este caso es conveniente codificar las diferencias sucesivas en lugar de las unidades de información. A pesar del ahorro que supone este sistema, presenta algunos inconvenientes:

- Realiza sumas y restas para codificar y decodificar los valores.
- Propaga errores
- Dotar al sistema de un mecanismo de arranque, ya que no existe la primera diferencia. Además el sistema debe de ser capaz de tratar las diferencias esporádicas mayores que el espacio reservado.

**Codificación por frecuencia de uso:** En número medio de bits se calcula mediante el sumatorio de todos los productos  $P_i \cdot C_i$ , donde  $P_i$  es la probabilidades de uso de la unidad de información y  $C_i$  es el número de bits asignado a la misma. Una propiedad importante que deben de cumplir estos códigos es que su tamaño debe estar implícito en el código.

## DESCRIPCION FORMAL DE UN COMPUTADOR

### 1-. Elementos funcionales.

Existen tres grupos de elementos funcionales, almacenamiento, operación e interconexión:

**Almacenamiento:** El elemento básico es el flip-flop o biestable. Un conjunto de biestables forman un registro. La longitud del registro expresa el número de bits que el registro es capaz de almacenar simultáneamente. El funcionamiento habitual es síncrono. La agrupación de un gran número de registros de igual longitud forma la memoria. El problema constructivo es el diseño de la lógica de control adecuada que permita seleccionar el registro entre los muchos que la forman. Lo que deriva en la lentitud de acceso a la misma.

Se denomina ancho de palabra a la longitud del registro elemental de la memoria. El tamaño en palabras de una memoria indica el número de registros individuales que la componen. La capacidad de una memoria viene determinada por el producto de su tamaño por el ancho de palabra. Al número que identifica biunívocamente una palabra o registro individual de la memoria se le conoce como dirección de memoria o posición de memoria. En términos generales se puede entender las señales de escritura y lectura como señales de sincronismo, puesto que marcan el instante de comienzo de su respectiva operación. El tiempo de ciclo mide la duración máxima de cada operación, establece el periodo de repetición de las operaciones de una memoria y mide de alguna manera la velocidad operativa de la misma. Es habitual que las memorias generen una señal de salida, denominada fin de ciclo, que marca el instante preciso en que la operación ha finalizado. La señal de fin de ciclo se activa al cabo de un ciclo completo desde que se activo la correspondiente señal de lectura o escritura.

**Operación:** Se denomina operador a todo circuito electrónico capaz de realizar una operación aritmética o lógica. Si es capaz de operar todos los bits de los operandos se dice que es un operador de tipo paralelo. Si sólo trabaja con un bit de cada operando a la vez, se tratará de un operados serie, que necesitará elementos auxiliares como registros con desplazamiento. Según el número de operandos pueden ser diádicos o monádicos. Según el ámbito de operación los operadores pueden dividirse en generales y especializados. Los operadores generales realizan distintos tipos de operación y los especializados una sola clase de operación (sumador / restador) .

**Interconexión:** Para transferir un dato entre dos elementos es preciso utilizar un camino o enlace entre ambos, con n canales o líneas para permitir la transferencia simultánea de un dato de n bits. Si un mismo elemento puede recibir información de más de un origen necesitará ciertas señales de selección. La forma más usual de interconexión de elementos en un computador es a través de un bus o cable. El bus es una vía de enlace a la que se puede acceder desde cualquiera de los elementos que se desean interconectar. Desde un punto de vista funcional un bus es un elemento cuya función es permitir una comunicación selectiva entre un conjunto de dispositivos conectados a él.

Los computadores utilizan normalmente tres buses: **datos**, cuyo ancho coincide con el ancho de palabra de memoria, **direcciones**, cuyo ancho está relacionado con el tamaño de la memoria y **control**.

### 2-. Estructura funcional básica de un computador.

Desde un punto de vista funcional la estructura básica de un computador consta de cuatro bloques o unidades fundamentales: la unidad de memoria, la unidad aritmético lógica, la unidad de entrada / salida y la unidad de control, conectadas entre sí por medio de buses.

**Unidad de Memoria:** Suele hacerse referencia a ella como memoria principal, está formada por un elemento de memoria al que se encuentran asociados dos registros auxiliares, registro de direcciones (RD) y registro de datos (RM). El registro de direcciones almacena temporalmente la dirección de memoria en la que se va a escribir o a leer un dato. El registro de datos almacena temporalmente el dato que se intercambia con la memoria, tanto en el caso de una operación de escritura como de lectura.

**Unidad Aritmético lógica:** Su elemento primordial es un operador diádico de propósito general. Cuenta con una serie de biestables (indicadores de resultado) que suelen estar agrupados con otros biestables de estado general del computador y conforman el registro de estado que se considera asociado a la unidad de control.

**Unidad de control:** Su misión fundamental es la de recoger las instrucciones que componen un programa, interpretarlas y controlar su ejecución.

**Unidad de Entrada /Salida :** Denominada unidad de intercambio de información se encarga de realizar las conexión y adaptación de la UCP con una gran variedad de dispositivos periféricos. En ella se distinguen dos elementos fundamentales:

- El controlador que gestiona directamente el dispositivo periférico y que guarda escasa o nula relación entre un dispositivo y otro.
- Interface: que es la parte que se encarga de gestionar el intercambio de información entre el dispositivo y la UCP.

Las funciones básicas de la Entrada/Salida son:

- 1- Seleccionar el dispositivo, lo que implica un mecanismo de direccionamiento de periféricos.
- 2- Disponer de un enlace entre la UCP y el dispositivo seleccionado que permita la transferencia en ambos sentidos.
- 3- Establecer un mecanismo de control de la transferencia riguroso que permita la sincronización o coordinación de la temporización de las operaciones de Entrada/Salida.

Las operaciones de entrada salida son responsabilidad tanto de la UCP como del propio periférico. El sistema a bus único utiliza los mismos buses del computador para gestionar la Entrad/Salida, asignándole direcciones y puertos. Con esta organización no existe distinción entre la memoria y los dispositivos de Entrad/Salida. La ventaja es la sencillez de estructura y la simplificación de las instrucciones, su inconveniente es que no permite la transferencia simultánea entre el procesador y la memoria y el procesador y los periféricos. Esto se soluciona mediante la estructura de bus dedicado, que incorpora un juego de buses diferentes para la Entrada/Salida que para la memoria. La interfase de Entrada/Salida tiene como componentes básicos son:

- **Decodificador de direcciones:** Permite reconocer al dispositivo su dirección al ser colocada en el bus de direcciones por la UCP.
- **Registro de datos:** Almacena temporalmente los datos intercambiados con la UCP a través del bus de datos de la Entrada/salida.
- **Registro de estado:** Es un conjunto de biestables de igual naturaleza que los registros de estado de la UCP.
- **Circuito controlador de interfase:** Se encarga de organizar las operaciones a realizar en el interior de la propia interfase, así como de coordinar la transferencia con la UCP.

Hay tres procedimientos básicos para provocar una operación de Entrada/salida:

- E/S controlada por programa: La ejecución desde la UCP de una instrucción de E/S desencadena la transferencia de una palabra de datos entre el computador y el periférico mediante una operación de E/S.
- E/S controlada por interrupciones: El propio periférico solicita al inicio de una operación de transferencia de E/S. Cuando el periférico se encuentra en disposición de realizar una transferencia, se lo indica a la UCP mediante una señal denominada solicitud de interrupción. La UCP suspende temporalmente la ejecución de las instrucciones en curso para ejecutar la rutina de servicio de interrupción que contiene la orden concreta de realizar la transferencia de E/S. Una vez efectuada la transferencia elemental, la UCP continúa el programador la instrucción en que lo suspendió, hasta que se produzca otra interrupción.
- E/S controlada por DMA: Cuando se tiene que efectuar una transferencia de un bloque de datos, la UCP informa a la interfase de DMA el origen en memoria a partir del cual empieza la transferencia, el tamaño total del bloque de datos a transferir, desentendiéndose de la operación, que a pasa ser responsabilidad de la interfase.



## 8 MODOS DE DIRECCIONAMIENTO (MD)

### 1- Introducción.

Un modo de direccionamiento es un procedimiento que permite determinar un operando, o la ubicación de un operando o una instrucción. Varios son los tipos de información susceptibles de ser direccionados. Se denomina objeto a la instrucción, operando o resultado que se desea direccionar, de forma que un objeto puede residir en memoria, en un registro o en la propia instrucción. La finalidad de los modos de direccionamiento es especificar el lugar concreto en que se encuentra el objeto. Unas consideraciones generales sobre los modos de direccionamiento son:

- Los programas utilizan varios modos de direccionamiento
- Un modo de direccionamiento puede utilizar diferentes registros como soporte de información, así como campos distintos del formato de la instrucción. En general, una instrucción deberá contener en su formato tantos campos de modo de direccionamiento como operandos utilice.
- Algunas arquitecturas, no permiten la utilización de determinados modos de direccionamiento.
- Cada modo de direccionamiento puede combinarse con los demás, de manera que el número de modos de direccionamiento es en teoría ilimitado.

### 2- Modos básicos de direccionamiento.

**Direccionamiento inmediato:** La instrucción contiene al objeto, que en este caso es un operando, es decir, la instrucción opera con valor constante.

**Direccionamiento Directo:** La instrucción contiene la dirección real del objeto

- **Direccionamiento absoluto:** La instrucción contiene la dirección exacta, sin compactas, en que se encuentra el objeto. En el modo de direccionamiento absoluto el objeto está en una posición de memoria principal. Sus características son:
  - No se precisan cálculos previos para conocer la dirección final de memoria.
  - Se necesita un ciclo de memoria más que en el inmediato para acceder al objeto.
  - Se puede conseguir mayor capacidad de direccionamiento utilizando formato de dos palabras, aunque añade una lectura de memoria adicional.

Se puede utilizar la denominación de direccionamiento de página base cuando el rango del modo de direccionamiento es inferior al mapa total de memoria.

- **Direccionamiento mediante registro:** El objeto no se encuentra en memoria sino en uno de los registros de la UCP. En la instrucción se indica de que registro se trata. Sus características principales son:
  - No precisa lectura adicional para disponer del objeto.
  - El rango del operador coincide con el del registro empleado.

**Direccionamiento relativo a registro:** La dirección exacta del registro no se encuentra en ningún lugar sino que ha de ser calculada. La instrucción contiene un campo del tipo CD un desplazamiento que hay que añadir a la dirección marcada por un puntero para obtener la dirección final del operando. El puntero suele estar almacenado en un registro determinado del computador. Su principal ventaja es la de poder direccionar todo el mapa de memoria con menos bits que los necesarios para toda la memoria direccionable. El único problema es de actualizar debidamente el registro puntero. Su inconveniente es que la dirección del objeto no se encuentra sino después de realizar una operación de suma.

- **Direccionamiento relativo al registro contador del programa:** Es este tipo se utiliza el registro contador del programa como puntero. Supone proximidad entre la zona de programa y la zona de objetos direccionables. Se suele emplear para direccionar instrucciones cercanas a la instrucción en curso, para realizar saltos relativos o bucles.
- **Direccionamiento relativo al registro base:** El puntero está en un registro que se denomina registro base. Este permite a los sistemas operativos multiusuarios reubicar con facilidad los programas de usuario de manera que no se produzca solapes entre unos y otros.
- **Direccionamiento relativo a pila:** El desplazamiento contenido en la instrucción se suma al valor del registro puntero de pila para obtener la dirección del objeto. El campo de desplazamiento suele ser bastante pequeño ya que en una pila los operandos suelen estar en los alrededores de la cima. Permite instrucciones muy compactas, debido a que la instrucción no requiere apenas información de dirección y se utiliza bastante en microprocesadores y minicomputadores.

**Direccionamiento indexado:** Un registro índice contiene la dirección de referencia y actúa de puntero. Se emplea para recorrer estructuras de datos del tipo vector o tabla. El registro mantiene un valor entero positivo, llamado índice, que indica la dirección de memoria del elemento concreto de la estructura que en cada momento está siendo recorrido. El mecanismo de direccionamiento es el mismo que en el relativo a registro base. Para el recorrido se permiten incrementos y decrementos de forma automática de una cierta magnitud, lo que da lugar a cuatro variantes:

- **Preincremento:** El registro índice se incrementa primero y luego su valor se suma al desplazamiento de la instrucción para obtener la dirección del objeto.
- **Predecremento:** El registro índice se decrementa primero y luego su valor se suma al desplazamiento de la instrucción para obtener la dirección del objeto.
- **Posincremento:** Primero se calcula la dirección del objeto sumando el desplazamiento con el valor del registro índice. A continuación se incrementa el registro.
- **Posdecremento:** Primero se calcula la dirección del objeto sumando el desplazamiento con el valor del registro índice. A continuación se decrementa el registro.

Otra diferencia significativa del direccionamiento indexado con respecto al relativo a registro base consiste en que el valor del registro índice se suele modificar con frecuencia en la ejecución de un programa, mientras que el del registro base no.

**Direccionamiento indirecto:** Conocido como indirección comienza con un direccionamiento directo, con el que se obtiene una dirección intermedia, que se utiliza para una nueva lectura en memoria para obtener el objeto. Sus características son:

- No se precisan cálculos previos para conocer la dirección final.
- Se necesitan ciclos de memoria para acceder al objeto.
- Permite una gran capacidad de direccionamiento al poderse utilizar todos los bits de la palabra de memoria como dirección.

Este direccionamiento es el más indicado para aplicaciones que utilizan datos situados en posiciones distantes de memoria, tales como la transferencia de los parámetros de un programa principal, subrutinas o montaje de enlaces en bancos de datos.

### **3-. Modos específicos de direccionamiento.**

**Registro indirecto:** Se identifica primero el registro utilizado, que contiene la dirección del objeto. Es equivalente al direccionamiento relativo a registro sin usar desplazamiento y pudiendo utilizar como registro de referencia uno cualquiera de propósito general, en lugar de registros específicos (pila, base o índice).

#### **Indirecto indexado:**

- **Posindexación:** Primero se interpreta la indirección y luego la indexación.
- **Preindexación:** Primero se interpreta la indexación y luego la indirección.

**Paginado:** Es una generalización del direccionamiento de página base. Considera el mapa de posiciones de memoria dividido en páginas. Cada página está formada por un número fijo de palabras, por lo que todas las páginas tienen el mismo tamaño. Para acceder a una determinada palabra se necesita concatenar dos campos, el primero, denominado contador o identificador de página que permite seleccionar la página; el segundo o campo de dirección de palabras, señala la posición de la palabra buscada dentro de la página. El encadenamiento de ambos campos se realiza en un registro de selección de palabra bajo la supervisión de la unidad de control. Previamente se chequea un bit especial que indique si el direccionamiento es de página cero y no se precisa concatenación de campos. Su principal desventaja es la de no permitir que una instrucción acceda a un operando situado fuera de la página direccionada. Si se utiliza un único registro indicador de página, para el código y para los datos, obliga a la proximidad entre instrucciones y los datos del programa. La paginación obliga a dividir los programas en módulos que quepan en trozos de páginas pero se debe impedir que un mismo módulo se solape entre dos páginas. Para evitarlo se usa la paginación con otro tipo de direccionamiento.

**Segmentado:** Se emplea en aplicaciones de programación modular, que utiliza zonas contiguas de memoria o segmentos, por cada función que realiza en programa. Los segmentos pueden estar ubicados en cualquier zona de memoria. Los distintos segmentos se pueden contabilizar mediante una tabla de segmentos. Esta tabla ha de mantener por cada segmento dos valores: un origen que define la dirección absoluta donde comienza el segmento y otro tope, que define la dirección final y delimita el tamaño del segmento. El inconveniente principal de la segmentación es el problema de los huecos de memoria que pueden quedar entre segmentos operativos. Para trabajar con direccionamiento segmentado la UCP ha de disponer de unos registros específicos, denominados registros de segmento. Suelen ser:

- *Registro de segmento de código:* Apunta al origen del segmento que almacena el programa que se ejecuta.
- *Registro de segmento de pila:* Apunta al origen del segmento que posee la zona de pila que se está utilizando.
- *Registro del segmento de datos:* Apunta al origen del segmento que contiene los datos del programa.
- *Registro de segmento complementarios:* Se utiliza para acceder a otro segmento de datos cuando el programa no tiene bastante con uno.

La dirección real se obtiene sumando la dirección de origen del segmento con el desplazamiento incluido en la instrucción y que normalmente se almacena el otro registro asociado al modo de direccionamiento segmentado.

## TIPO Y FORMATO DE INSTRUCCIONES

### 1- Introducción

El lenguaje máquina está ligado a la construcción interna del computador, y es consecuencia de la estructura física del mismo, existiendo una incompatibilidad innata entre distintos computadores. Las propiedades que suelen cumplir las instrucciones de máquina son las siguientes:

- Realizan una única función.
- El número de operandos de cada instrucción es fijo, adaptándose a formatos predefinidos.
- La codificación de las instrucciones es sistemática, normalmente por campos.
- Las instrucciones son autocontenidas e independientes, es decir su interpretación no depende ni de otras instrucciones ni de la posición que ocupen en el programa o memoria.

La información mínima que ha de contener una instrucción es:

- a- Operación a realizar
- b- Localización de los operandos
- c- Localización del resultado de la operación
- d- Identificación del tipo de operandos de la operación
- e- Situación de la siguiente instrucción.

La definición del juego de instrucciones de un computador obliga a considerar los siguientes puntos:

- El conjunto de operaciones realizadas por el computador.
- Representación o representaciones de los datos manejados.
- Modos de direccionamiento de que se dispone el computador.
- Formato o formatos empleados para codificar las instrucciones.

### 2- Tipos de instrucción. Clasificación.

Desde un punto de vista formal el juego de instrucciones de un computador debe ser completo y eficaz. La completitud se refiere a que se pueda calcular con él en un tiempo finito, cualquier tarea computable. La eficacia implica conseguir un alta velocidad de cálculo sin complicar la UAL ni la Unidad de Control. Aunque cada procesador tiene su conjunto de instrucciones de máquina, todos los lenguajes poseen similitudes en la estructura y concepto. Lo que ha llevado a esclarecer clasificaciones de las instrucciones más frecuentes según diferentes puntos de vista. Una organización debida a Fairclough establece una división en ocho grupos:

- 1- **Movimiento o transferencia:** Permiten copiar el contenido del operando origen en el operando destino. No afectan a los biestables.
- 2- **Modificación de la secuencia del programa, saltos:** Provocan un salto hasta una instrucción que no es la siguiente. Se puede considerar como de transferencia en el sentido de que el destino es el contador de programa. Pueden ser de salto condicionado o salto con retorno.
- 3- **Instrucciones aritméticas:** Efectúan alguna de las cuatro operaciones aritméticas fundamentales sobre alguno de los posibles datos numéricos. Afectan a los biestables de estado.
- 4- **Comparación:** No se realizan para almacenar un resultado sino para afectar a los biestables de estado y combinarse con una instrucción de salto condicionado.
- 5- **Lógicas:** Se realizan de manera independiente en cada uno de los bits que componen los operandos. El resultado afecta a los biestables de estado.
- 6- **Desplazamiento:** Permiten procesar por partes la información contenida en un dato.
- 7- **Manipulación de bits:** Realizan todas las operaciones booleanas posibles sobre los bits que componen un determinado campo de una posición de memoria o registro, afectando a un bit en particular.
- 8- **Entrada y salida. Misceláneas.** Las instrucciones de E/S son de transferencia en las que el origen o el destino son registro del periférico.

### 2- Formato de instrucciones

El formato define la longitud o número de bits que la componen, así como el significado de cada uno de ellos. El formato de una instrucción se divide en campos cada uno de los cuales contiene una información específica. Los dos campos básicos en un formato son el código de operación, que indica la operación a realizar, y el campo de dirección, que determina la dirección de un dato, resultado o instrucción a la que hay que bifurcarse. Las características generales que deben reunir los formatos de instrucciones elegidos para un computador son:

- Un computador debe poseer uno o unos pocos formatos en los que debe acomodar todas sus instrucciones, para simplificar la Unidad de Control.

- Los formatos han de ser sistemáticos: Los campos del mismo tipo tienen la misma longitud y, para un mismo formato, están siempre en la misma posición, lo que simplifica la decodificación.
- Uso de direccionamientos implícitos para reducir el tamaño del campo de dirección.
- El tamaño de los formatos suele amoldarse a la longitud de palabra de memoria del computador.

**Campo de código de operación:** La misión de este campo es la de identificar la operación que realiza cada instrucción. El campo de código de operación no es opcional. Sus características más importantes son:

- El código de operación implica habitualmente el tipo de operando ya que es normal no emplear un campo específico para indicar la representación de los operandos. Por ello se emplean códigos de operación distintos, para diferenciar una misma operación que trabaje con distintos tipos de datos.
- Cuando un computador utiliza más de un formato, éstos se distinguen por el código de operación que siempre es el primer campo del formato.
- El tamaño del campo de código de operación es fijo para todos los posibles formatos de un computador.

**Campo de operando y resultados:** Las instrucciones pueden contener campos de dirección para identificar las posiciones de los operandos con los que operan, así como la posición del resultado. Por ello hay instrucciones con un, dos o ningún campo de dirección. Los campos de dirección pueden encontrarse divididos en subcampos. Los más significativos son:

- Modo de direccionamiento. Es un campo donde se codifica el modo de direccionamiento a emplear para localizar el dato .
- Campo de registro. Sirve para direccionar un registro de la UCP que puede contener el operando, o la dirección del mismo, o ser el registro base para un direccionamiento relativo, etc., según el modo de direccionamiento utilizado.
- Dirección. Puede contener una dirección absoluta, un desplazamiento relativo, un operando inmediato, etc. Dependiendo del modo de direccionamiento empleado.

El tamaño de los campos de direcciones absolutas deben de corresponderse con los mapas direccionados . Esto no afecta a los campos de desplazamiento, que no tienen que abarcar todo el mapa de memoria ya que la dirección final se compone con la ayuda de un registro de la UCP.

**Campo de condición:** En las instrucciones de salto condicional se ha de codificar de alguna forma la condición ante la cual el salto ha de producirse, siendo habitual el empleo de un campo específico para la condición. Existen varias alternativas, una de ellas es la de emplear un campo formado por una máscara binaria de condición. La máscara contiene un bit por cada uno de los biestables de estado que pueden utilizarse en la condición, y puede contener un bit por cada indicador complementado. El criterio que se emplea es el de poner a 1 los bits de la máscara correspondientes a los biestables que se deben examinar, dejando a 0 los demás. En las instrucciones de salto condicional, la UCP utiliza esta máscara para contrastarla con los valores actuales de los biestables de estado, con los que realiza operaciones lógicas. Si el resultado es cierto se produce el salto, sino se sigue la secuencia normal.

Otra alternativa es la codificación directa para el campo de condición. En este método se emplea un código binario distinto para cada una de las condiciones contempladas por las instrucciones de salto. La unidad de control necesita entonces realizar una decodificación previa para determinar los biestables de estado implicados en cada condición.

## PERIFERICOS

### 1- Introducción

Existen diversos criterios a la hora de clasificar los periféricos. Se puede distinguir entre locales y remotos, atendiendo a la distancia que los separa de la UCP. Otra clasificación se basa en el soporte utilizado por los periféricos para mantener la información. Otra clasificación obedece a la función que desempeñan respecto a un computador y se distingue entre:

- Dispositivos de entrada: Capaces de suministrar o introducir información al sistema computador.
- Dispositivos de salida: Sacan al exterior los resultados obtenidos en el procesamiento.
- Dispositivos de entrada/salida: Encargados de introducir y extraer información del sistema.
- Dispositivos de comunicación con otros computadores: Establecen intercambios de información entre distintos computadores.
- Dispositivos de comunicación con sistemas físicos: Encargados de facilitar el trasiego de información entre un computador y un determinado sistema con objeto de supervisarlos y controlarlos.

### 2- Periféricos de entrada.

Lectora de tarjetas, banda perforada, teclado, ratón, Scanner, tableta digitalizadora, lápices ópticos, lectores de tinta magnética (MICR), lectores de marcas ópticas, lectores de códigos de barras, reconocedores de voz, palanca (Joystick), pantallas sensibles al tacto, etc.

### 3- Periféricos de salida

**Tarjetas controladores y monitores de vídeo:** Los monitores de vídeo se pueden clasificar en:

- Monocromo: fósforo verde o ámbar, trabajo con texto
- Color

Atendiendo a las posibilidades de representación:

- De pantalla alfanumérica
- De pantalla gráfica

Las características de los monitores son: silenciosos, rápidos, reutilizables, económicos, bidireccionales si se utilizan con el teclado. Además tienen como prestaciones adicionales : el enrollado (scrolling), doble página, pantalla partida, atributos de caracteres ( vídeo inverso, parpadeo, subrayado) , control del cursor.

Otro grupo de propiedades de los monitores son sus características generales entre las que se encuentran: Tamaño de la pantalla, capacidad alfanumérica ( 24 líneas × 80 caracteres), capacidad gráfica, relacionada con la resolución ( número de puntos individualizados [pixels] que es capaz de representar), conexión al computador.

**Tarjetas controladoras:** Su función es preparar la información que en cada momento deberá visualizarse y generar las señales analógicas de ataque a los circuitos electrónicos del monitor. El volumen de información que maneja la tarjeta depende del tipo de monitor que vaya controlar, esta información se encuentra codificada en la memoria de vídeo de la tarjeta que deberá ser mayor o menor según el tipo de monitor con el que se vaya a usar. El tipo de codificación empleada es arbitrario si bien, se puede establecer unos criterios que permiten cuantificar la cantidad de memoria necesaria.

**Funcionamiento en modo texto:** Se emplea en pantallas monocromas, por lo general todos los caracteres representables se encuentran almacenados en el generador de caracteres. Por cada carácter se tiene una matriz de puntos indicando cual debe ser iluminado y cuales no ( se requiere un bit por punto de matriz).

Memoria requerida = caracteres/palabra × bit/carácter = bits

**Funcionamiento en modo gráfico:** La información a almacenar es por cada punto en lugar de por carácter. La memoria de vídeo se considera en planos, contiene cada plano 1 bit por cada punto. Una memoria de 4 planos contiene 16 combinaciones por punto, pudiendo representar por ejemplo 16 colores distintos. Esta organización permite calcular fácilmente la capacidad de memoria necesaria en la controladora a partir de los puntos del monitor:

Memoria requerida = puntos × planos/punto × bit/plano = bits

El generador de barrido tiene como misión leer la memoria de vídeo y transformar la información que posee de cada punto en la señal analógica adecuada para gobernar el tubo de rayos catódicos.

**Pantallas de cristal líquido:** Tienen como característica principales la relación de contraste, que es la diferencia de brillo entre zonas claras y oscuras ( 1 .. 100); la velocidad de refresco, tiempo que tarda la pantalla en

refrescar la información sin producir estelas ni imágenes manchadas; tamaño; consumo; relación de aspecto: ancho y alto; peso.

**Impresoras:** Se clasifican por el tipo de papel, forma de imprimir o el mecanismo de impresión.

**Reproductoras de microfichas:** Periféricos de salida que utilizan como soporte de información un material fotosensible.

**Trazadores de gráficos:** Plotters. **Sintetizadores de voz, hologramas, etc.**

#### **4- Periféricos de Entrada/Salida**

**Discos magnéticos:** Son dispositivos de almacenamiento magnético de acceso directo. El disco tendrá tantas pistas como paradas en posiciones fijas puede realizar el brazo que contiene la cabeza lectora y que se desplaza en dirección radial hacia el centro del disco. La información se graba en cada pista en paquetes completos de datos que constituyen sectores. Dado que el número de sectores es el mismo en todas las pistas, la densidad de grabación es mayor hacia el centro que hacia la periferia.

**Discos duros:** Existen tres categorías: discos duros encapsulados en un cartucho de plástico circular que son intercambiables; discos múltiples apilados sobre un eje común llamados de pila o torre de discos, también removibles y enfundables en plástico transparente y discos que se colocan en recintos herméticos y no son intercambiables (Winchester).

El número de platos en los discos de tipo pila es variable, oscilando entre 3 y 12. Normalmente de cada plato se usan ambas caras, excepto la superior del más alto y la inferior del más bajo, que no se usan para evitar problemas de suciedades y raspaduras. Los brazos que contienen las cabezas lectoras/grabadoras se articulan en un único peine que mueve todas las cabezas a la vez en el interior de la pila. Para una posición dada del peine hay tantas pistas direccionadas simultáneamente como cabezas tiene el disco. A este conjunto de pistas se le denomina cilindro; y hay tantos cilindros como pistas tiene cada plato individual.

**Discos ópticos:** Son dispositivos de almacenamiento en que los bits se encuentran fijados en forma de huecos microscópicos producidos por un rayo láser, que quema pequeños puntos de la superficie. Presenta como ventajas la gran compactación, el acceso directo, la alta velocidad de transferencia y el bajo costo. Su inconveniente es la vulnerabilidad al polvo y falta de borrado y reescritura.

#### **5- Comunicación con otro computador**

Si los computadores se encuentran a pocos metros de distancia se puede emplear la transmisión modo paralelo. Si están situados a más de 10 metros hay que recurrir a una línea o canal físico diferente, y suele usarse una onda portadora sinusoidal (analógica) para modular la información. En este caso se transmite en modo serie. La línea de comunicación puede ser explotada de diversas maneras. Si los datos sólo pueden viajar en un sentido es una línea simplex. Si pueden viajar en ambos sentidos será una línea duplex. Puede tratarse de una línea semiduplex ( Halfduplex ) si los datos viajan en un sentido primero y luego en el otro o duplex total ( Fullduplex ) si la transmisión es simultánea en ambos sentidos.

El módem es un dispositivo para la comunicación entre computadores que emplea las ventajas de la transmisión sinusoidal. La señal digital se convierte en analógica mediante un proceso de modulación y la señal analógica se transforma en digital mediante un proceso de demodulación.

#### **6- Comunicación con un sistema físico.**

Para esta conexión se precisan los denominados periféricos de control.

**Sistemas de lectura analógica:** Está basado en la conversión de la señal analógica en su correspondiente valor numérico, para que pueda ser entendida por el computador. Este sistema se compone de:

- Sensor: convierte la señal de entrada a medir en una señal de tipo eléctrico.
- Conformador de señal : Filtra, amplifica y linealiza la salida del sensor, y transmite la señal desde donde se produce hasta donde se encuentra el resto del equipo.
- Etapa de aislamiento: Produce un aislamiento galvánico entre el computador y el sistema físico.
- Convertidor analógico/digital: Toma corriente o tensión y realiza su conversión a valor numérico, devolviendo un resultado en binario. Mediante un conmutador o multiplexor analógico se conecta a su entrada las diversas señales a medir.

**Sistema de accionamiento analógico:** Es el contrario del anterior, además del conformador de señal y etapa de aislamiento consta de:

- Convertidos Digital/Analógico: Convierte una cantidad numérica enviada por el computador en una tensión o corriente.
- Accionador: Convierte la señal estándar del convertidos en la señal que requiere el sistema físico.

Es este sistema se debe destacar la necesidad de que la señal generada esté de forma continuada en la salida durante todo el accionamiento a diferencia del sistema de lectura analógica que señaliza sólo en instantes concretos en los que interesa.

**Sistema de lectura digital:**

- Etapa conformadora: Convierte la señal digital del sistema físico en una señal de nivel TTL con la que es capaz de trabajar el computador.
- Etapa de aislamiento: Son octo-acopladores que aíslan galvánicamente las señales del sistema físico de las del computador.
- Buffers : recogen varias señales digitales y las envían al computador.

## PRESTACIONES Y APLICACIONES

### 1- Prestaciones típicas de un computador

- Ancho de palabra: Indica el número de bits que procesa en paralelo el computador.
- Precisión y rango: Como precisión se entiende en este caso el número de dígitos significativos que posee la información. Está directamente relacionada con el ancho de palabra. El rango indica el conjunto de valores numéricos que se pueden representar.
- Capacidad de memoria: RAM y disco duro.
- Velocidad del computador: Viene dada por el número de instrucciones que se pueden ejecutar por segundo. La unidad usual es el MIP (Millón de Instrucciones por segundo) Una alternativa es el MFLOP ( Millón de Operaciones en coma flotante por segundo ), más relacionada con la capacidad de cálculo científico. Los MHz indican los millones de ciclos de reloj interno ejecutados en un segundo.
- Tiempo de acceso a memoria principal y periféricos: El ancho de banda de la memoria indica la velocidad de acceso de un computador a su memoria y que técnicamente consiste en el número de palabras a las que el procesador puede acceder en un segundo. Esta velocidad se expresa en megabytes por segundo o megapalabras por segundo. Influye decisivamente en el rendimiento del computador. La velocidad de acceso a periféricos también se debe tener en cuenta para evaluar el rendimiento de un sistema computador.
- Índices de rendimiento: Unos pequeños programas que ejecutan aplicaciones típicas dan como resultado el índice de rendimiento, que es un número que permite comparar el rendimiento de distintos computadores.

### 2- Tecnologías avanzadas.

La arquitectura Von Newman se caracteriza por ser de tipo escalar o secuencial, es decir, existe un único flujo de instrucciones y un flujo único de datos. La alternativa a esta arquitectura consiste en buscar máquinas de funcionamiento en paralelo, lo que permite mejorar la potencia del computador. Existen dos modos básicos de usar el paralelismo en los computadores:

**Paralelismo interno o implícito:** Queda oculto en la arquitectura del computador. Se implanta a nivel de la estructura interna del computador, y afecta a la construcción de la máquina. Sirve para aumentar la velocidad del computador sin modificar su funcionamiento básico. La técnica más utilizada es la segmentación o pipe-line, que se basa en dividir una función en varias subfunciones o etapas que se realizan independientemente. Los computadores cuyas arquitecturas utilizan este principio reciben el nombre de computadores segmentados o computadores vectoriales, ya que el dispositivo encargado de la segmentación recibe como datos, estructuras que son vectores.

**Paralelismo externo o explícito:** Queda visible al usuario, siendo este el encargado de explotarlo de forma óptima. Las máquinas explícitamente paralelas pueden clasificarse según el número de flujos de instrucciones y de datos que circulan por el computador. El flujo de instrucciones está formado por el conjunto de instrucciones que se van leyendo en la ejecución de un programa, mientras que el flujo de datos está formado por los datos que dichas instrucciones van leyendo y almacenando. Esta clasificación considera los cuatro casos posibles:

- Arquitectura SISD (Single Instruction Single Data) : Arquitectura serie con flujo de instrucciones y otro de datos, que se corresponde con la arquitectura clásica de Von Newman.
- Arquitectura SIMD ( Single Instruction Multiple Data ): Arquitectura con un sólo flujo de instrucciones y varios flujos de datos. Consta de una única unidad de control que gobierna un número determinado de unidades de procesamiento, todas ellas iguales. Cada una de estas unidades posee su propia memoria local, sus registros y su unidad aritmético-lógica. El mecanismo que controla la circulación entre el procesador y

la memoria principal recibe el nombre de red de conexión. Los procesadores matriciales, con un rendimiento superior a los escalares son de esta arquitectura.

- Arquitectura MISD ( Multiple Instruction Single Data) : Arquitectura con varios flujos de instrucciones y uno de datos. Puede considerarse como una superestructura pipe-line formada por varios procesadores, en la que cada uno hace parte del trabajo sobre un flujo continuo de datos que circula entre ellos.
- Arquitectura MIMD (Multiple Instruction Multiple Data): Arquitectura multiprocesadora con varios flujos de instrucciones y otros tantos de datos. Consiste en dotar al computador de varios procesadores, cada uno con su propia unidad de control y su juego de instrucciones, de modo que sean capaces de realizar acciones independientes sobre distintos conjuntos de datos.

Una de las características fundamentales de los multiprocesadores es el nivel de acoplamiento entre los procesadores que lo componen, distinguiéndose tres niveles básicos:

- Acoplamiento débil: Varios computadores serie se comunican entre sí a velocidades altas, mediante una red local.
- Acoplamiento moderado: Distintos procesadores gestionados por un único sistema operativo, y con un único mapa de memoria que sirve como elemento de comunicación entre los procesadores. El acceso de los distintos procesadores a los recursos del sistema no es homogéneo.
- Acoplamiento fuerte: Todos los procesadores del sistema pueden utilizar todos sus recursos. La memoria principal también es común y sirve para comunicar los distintos procesadores entre sí.

Un caso especial de arquitectura MIMD lo constituyen los computadores sistólicos, compuestos por un número elevado de procesadores interconectados por una red fija.

#### **Tecnologías avanzadas de memoria:**

- Memorias entrelazadas: Mejora la velocidad de acceso a la memoria principal. Consiste en dividirla en módulos autónomos. Cada uno de ellos está dotado de funcionamiento independiente, por lo que se puede acceder simultáneamente a tantas posiciones de memoria principal como módulos tenga esta. Permite ir entremezclando los accesos entre los distintos módulos. Cuando surgen peticiones simultáneas sobre el mismo módulo, aparece una colisión, por lo que se debe atender una y hacer esperar las otras.

- Memoria caché: Es una memoria auxiliar de alta velocidad que se añade a la memoria principal para acelerar su funcionamiento. Se sitúa entre el procesador y la memoria principal, se alimenta de bloques de datos procedentes de la memoria principal y reduce substancialmente los tiempo que debe esperar el procesador entre que realiza las peticiones de datos y los recibe de memoria. El mecanismo de acceso es: Primeros se realiza una prebúsqueda en la memoria caché; sino se encuentra el dato se busca en la memoria principal. Los bloques almacenados en la memoria caché se mantendrán para una nueva y próxima utilización; o se sustituyen por otros si llevan un periodo largo sin ser utilizados.

- Memoria virtual: Un computador emplea memoria virtual cuando las direcciones que generan sus programas se refieren a un espacio mayor que el espacio realmente disponible de memoria principal. En estos computadores se debe distinguir entre el mapa de direcciones lógicas o virtual y el mapa de direcciones físicas o reales. El espacio virtual emplea como soporte un dispositivo de almacenamiento externo, mientras que el espacio físico se refiere a la memoria principal. Las razones principales del empleo de esta técnica son:

- Ofrecen al programador una máquina de memoria casi ilimitada.
- Permite establecer mecanismos de protección de memoria.
- Permite realizar una buena gestión de la memoria principal.

El solapamiento u overlay es un mecanismo que permite ejecutar programas de mayor tamaño que la memoria disponible. Para ello el programador divide su programa en bloques que se preparan para ejecutar en la misma zona de memoria. El propio programa tiene las instrucciones de E/S para traer en cada momento esos bloques de la memoria externa. Además de su complejidad este método tiene el inconveniente de hacer al programa dependiente de la memoria del computador, por lo que las variaciones en su configuración pueden obligar a revisar la división realizada en el programa.

Existen dos formas básicas de dividir tanto la memoria física como la lógica en bloques regulares de  $2^p$  palabras de tamaño. Cada bloque recibe el nombre de página y su dirección de comienzo es múltiplo de  $2^p$ . La segmentación



permite dividir la memoria en bloques de cualquier tamaño, que se denominan segmentos. Los segmentos constituyen unidades de cualquier tamaño, que se denominan segmentos. Los segmentos constituyen unidades lógicas desde el punto de vista del software, lo que presenta ventajas desde el punto de vista de proteger y compartir información.

- **Memorias asociativas:** La memoria principal que se utiliza es del tipo RAM. Esta memoria se caracteriza porque su contenido se referencia secuencialmente mediante la especificación de su dirección. Una alternativa a este tipo de memoria es la denominada memoria asociativa. Este tipo de memoria se direcciona por contenidos, permitiendo acceso paralelo a múltiples palabras de memoria. Las memorias asociativas se conocen como memorias direccionables por contenido, memorias de búsqueda paralela y memorias multiacceso. Su principal ventaja es su capacidad para realizar operaciones de búsqueda y comparación en paralelo. Su principal inconveniente es el coste del hardware. Se pueden encontrar algunos computadores SIMD con este tipo de memoria, y con los procesadores asociativos.

**Computador RISC:** La tercera forma de mejorar el rendimiento consiste en optimizar el juego de instrucciones. Las instrucciones normalmente incluyen funciones y operaciones complejas. Ello ha sido motivo de varias razones:

- La utilización de la microprogramación permite gran flexibilidad en el diseño del juego de instrucciones.
- La velocidad de acceso a memoria es menor que el tiempo de cálculo.
- Un amplio juego de instrucciones facilita el diseño de compiladores.

Los avances tecnológicos recientes han hecho que la velocidad de acceso a memoria principal haya aumentado, con lo que el tiempo requerido para leer una micro instrucción sea igual al de lectura de una instrucción, reduciendo el interés de usar la microprogramación. El empleo de un juego reducido de instrucciones simples permite reducir costes, aumentar velocidad, y facilita el diseño de compiladores según nuevas técnicas. Un computador RISC se diseña de acuerdo con los siguientes principios:

- Las funciones que realice el computador deben de ser lo más simples posibles.
- No debe emplear la microprogramación.
- La simplicidad de la decodificación de las instrucciones y el empleo de las técnicas pipe-line son los factores que se deben explotar en el diseño para hacer que la máquina sea rápida. Por ellos las máquinas RISC utilizan generalmente un formato único de instrucción, que suele ser de 32 bits.
- Las técnicas de compilación deben buscar la optimización mediante el uso de instrucciones sencillas en vez de complejas.

### **3- Aplicaciones**

- Para el cálculo científico y técnico.
- Interacción con el entorno físico
  - Tratamiento y procesamiento de señales
  - Control por computador . Consiste en:
    1. Medir el comportamiento del proceso.
    2. Compararlo con el deseado generando una señal de error
    3. En base a la señal de error generar una señal de actuación de acuerdo a una ley correctora
    4. Actuar sobre el proceso.

## INTRODUCCION AL LENGUAJE ENSAMBLADOR

### ***1- Concepto de lenguaje de programación.***

Desde un punto de vista informático un lenguaje es una notación formal para describir algoritmos o funciones que serán ejecutadas por un computador. Los lenguajes simbólicos de programación permiten usar códigos de operaciones mnemotécnicas, nombres simbólicos para las variables y las direcciones de memoria, y otras comodidades que dependen de cada lenguaje concreto. El empleo de lenguajes simbólicos obliga a disponer de un traductor, del lenguaje elegido al lenguaje de máquina.

El nivel más bajo de los lenguajes simbólicos lo constituyen los lenguajes ensambladores, en los que e tiene exactamente el lenguaje de máquina tal cual, solo que usando símbolos mnemotécnicos para los códigos de las instrucciones y para las direcciones de memoria. A cada sentencia del lenguaje ensamblador le corresponde una única instrucción de lenguaje máquina.

El segundo nivel de lenguaje simbólico está constituido por los llamados lenguajes de macros, o macroensambladores. El juego de instrucciones se divide en dos clases, la de las instrucciones normales y las denominadas macroinstrucciones, en las que a cada una le corresponde un conjunto preestablecido de instrucciones máquina.

El tercer nivel de lenguajes simbólicos es el de los lenguajes orientados al problema, que prescinden del lenguaje máquina particular del computador ( lenguajes de alto nivel).

Un cuarto nivel de lenguaje simbólico lo constituyen los lenguajes diseños de sistemas de información. Permiten definir directamente sistemas de información prescindiendo de los programas necesarios para llevar a cabo el diseño global. Los programas son generados automáticamente por el computador atendiendo a los requisitos de diseño que se le solicitan.

### ***2- Lenguajes orientado al problema.***

Un lenguaje de alto nivel es un método conveniente y sencillo de escribir las estructuras de información y las secuencias de acciones necesarias para ejecutar una tarea concreta. Se pueden establecer una serie de funciones que todo lenguaje de programación de alto nivel debe cumplir:

- Debe permitir especificar los objetos que el programador quiere que el computador manipule.
- Debe permitir especificar las operaciones que el programador desea que se apliquen sobre dichos objetos.
- Debe ser concebido de forma que pueda ser traducido automáticamente, sin intervención humana, al lenguaje en que debe ser explotado, lo que obliga a que cada operación pueda descomponerse en una secuencia de instrucciones elementales que el computador sea capaz de ejecutar.

Un lenguaje de alto nivel es independiente de la arquitectura del computador (en teoría), lo que presenta dos ventajas:

- La persona que desarrolla el programa no tiene que saber nada acerca del computador que ejecutará el programa.
- Los programas son transportables.

El inconveniente es el de tener que disponer de un traductor por cada máquina distinta en que se pretenda usar el mismo lenguaje de alto nivel. En la práctica las características de dependencia de la máquina persisten en la mayoría de implementaciones. Otra propiedad de los lenguajes orientados a problemas es que son más cortos que su equivalente en bajo nivel. Todo lenguaje de alto nivel persigue un conjunto de objetivos:

- Estructura clara : La sintaxis o regla gramatical de un determinado lenguaje define la estructura de los programas escritos en él.

- **Simplicidad:** Se logra con un reducido conjunto de operaciones binarias, unas cuantas reglas para combinarlas y la ausencia total de excepciones a estas reglas.
- **Eficiencia.**
- **Legibilidad:** Muchos lenguajes permiten la inclusión de comentarios, sin embargo un buen lenguaje de alto nivel debería permitir desarrollar programas legibles sin necesidad de comentarios adicionales.

En una primera instancia los lenguajes de alto nivel se pueden clasificar en lenguajes de propósito general y lenguajes de propósito específico. Los lenguajes de propósito general se emplean en los negocios, aplicaciones científicas o resolución de problemas de ingeniería, por lo que estos lenguajes son largos y relativamente difíciles. Las categorías más comunes de los lenguajes de propósito específico son los lenguajes comerciales, científicos y educativos.

Otra clasificación divide a los lenguajes en procedimentales y declarativos. Los lenguajes procedimentales establecen como debe ejecutarse una tarea partiéndola en procedimientos que especifican cómo realizar cada una de las tareas. Los lenguajes declarativos describen las estructuras de datos y las relaciones entre ellos que son significativas, para ejecutar una determinada tarea, al tiempo que indican cuál es el objetivo de dicha tarea.

### **3- Lenguajes de bajo nivel.**

Se encuentran totalmente vinculados al hardware, es decir, a la estructura del computador. Bajo esta denominación se engloba tanto el propio lenguaje máquina del computador como el lenguaje ensamblador.

**Lenguaje máquina:** Una instrucción en lenguaje máquina es una cadena de ceros y unos que expresan en forma codificada, las únicas ordenes que un computador es capaz de entender. Sus principales inconvenientes son:

- El programa debe expresarse en términos de instrucciones muy elementales y poco potentes; se requieren muchas instrucciones para una operación sencilla.
- Es necesario que el programador lleve el control explícito de la dirección concreta de memoria donde cargará cada instrucción del programa, y las direcciones de cada dato y resultado previstos.
- En las bifurcaciones el programador deberá expresar exactamente la dirección de memoria de la instrucción a la que se bifurcará.
- Para realización de un programa se debe consultar en todo momento la tabla de códigos octales o hexadecimales correspondientes a cada instrucción.

**Lenguaje ensamblador:** Surge con la idea de evitar las dificultades del lenguaje máquina, su misión es la de simplificar la programación de un determinado computador, manteniendo un control directo sobre el hardware del mismo. Puede considerarse como un lenguaje cuyas estructuras de datos corresponden con las estructuras físicas de los registros de la memoria principal del computador para el que está pensado, y cuyas instrucciones tienen una relación directa y biunívoca con las instrucciones del lenguaje máquina. La simplificación supuesta por los programas ensambladores proviene de dos aspectos:

- Empleo de código mnemotécnico para representar las instrucciones.
- Empleo de nombre simbólicos para designar los datos y las referencias.

Los lenguajes ensambladores presentan otra propiedad que los distingue de los lenguajes orientados a problemas: el programador en ensamblador tiene acceso a los recursos e instrucciones de la máquina a la que se aplica.

### **4- Edición de programas.**

El primer paso del desarrollo de cualquier programa consiste en escribir el código fuente, que en definitiva no es más que un texto escrito. Se suelen clasificar en editores de línea o editores de página.

### **5- Traductores de lenguaje.**

Su objetivo principal es el de convertir un programa escrito en un lenguaje simbólico determinado a su equivalente en lenguaje máquina. Un segundo objetivo es el de identificar los posibles errores del programa original. El programa traductor produce mensajes de diagnóstico para ayudar a corregir los errores.

**Programas fuente y objeto:** Un traductor es un programa escrito en un cierto lenguaje, que se conoce por lenguaje de implementación o lenguaje base, y que tendrá como entrada un texto escrito en lenguaje fuente y produce otro texto escrito en lenguaje objeto.

El lenguaje fuente es el lenguaje simbólico empleado para codificar y representar un problema. El lenguaje objeto es un código dispuesto para su ejecución en un determinado computador o que requiere unas fases previas (montaje y carga) para poder ser ejecutado.

**Compiladores e intérpretes:** Un compilador traduce el código fuente generando un programa en lenguaje máquina que contiene el código objeto equivalente. El proceso de compilación se aplica al código fuente tratándolo como un todo, y debe estar terminado de editar para poder compilarlo.

En un interprete cada una de las sentencias del programa fuente se analizan y ejecutan por separado; a medida que el programador va completando sus instrucciones el interprete las va analizando, diagnosticando los posibles errores que la instrucción puede traer, si no tiene ningún error, se ejecuta a continuación.

**Programa ensamblador:** Es un programa que traduce un texto escrito en el lenguaje ensamblador de un determinado computador al lenguaje máquina de ese mismo computador, y proporciona las facilidades necesarias para simplificar la tarea de desarrollar programas en lenguajes de bajo nivel.

Un objetivo más general es el de proporcionar una interfaz adecuada entre el programador y la arquitectura del computador, para las tareas de programación. El programa ensamblador no debe ocupar demasiado espacio y debe desarrollar las tareas recomendadas de forma rápida y eficiente.

## ***6- Programas cargadores.***

Para que un programa almacenado en un disco se pueda ejecutar debe ser cargado en la memoria principal del computador. Esta es la tarea que lleva a cabo el programa cargador que debe estar previamente en la memoria del computador.

La ejecución del cargador efectúa la secuencia adecuada de operaciones de E/S, necesarias para transferir un programa en lenguaje máquina desde una ubicación determinada del disco hasta una posición específica de la memoria principal. Una vez cargado el código objeto, el cargador comienza a ejecutar el programa objeto que acaba de depositar en memoria, haciendo un salto a su primera instrucción.

A fin de transferir un programa objeto en disco a la memoria principal, el cargador debe conocer la longitud del programa y su dirección inicial en la memoria. Esta información se coloca, por lo general, en el encabezamiento del fichero en disco que contiene al código objeto. Un cargador que transfiere un código objeto almacenado con este formato se conoce como cargador binario absoluto.

En muchos sistemas se precisa un cargador más complejo, el cargador reubicador. La razón es que un ambiente operativo normal para muchos computadores, consiste en mantener varios programas simultáneamente en memoria principal. Estos programas son de diferente tamaño, por lo que es deseable contar con cierta flexibilidad a la hora de determinar donde se coloca cada programa en el momento de cargarlo.

En resumen un programa cargador realiza tres funciones:

- Copia el código objeto en las posiciones de memoria donde residirá durante su ejecución.
- Reasigna las direcciones contenidas en el programa para que se ajusten a las direcciones en las que ha sido cargado.
- Lanza la ejecución del programa recién cargado.

## ***7- Programas montadores.***

El producto de un compilador o ensamblador puede ser un conjunto de módulos independientes en código máquina, que se corresponden con subprogramas o módulos parciales del programa fuente. Todos estos segmentos suelen expresarse en código reubicable, que no contienen sino direcciones relativas y, que no pueden ejecutarse hasta que se hayan sustituido por direcciones absolutas.

La misión del programa montador de enlaces es la de proporcionar las direcciones adecuadas para todas las instrucciones de llamada y retorno, de modo que los módulos queden correctamente enlazados entre sí, generando lo que se llama módulo ejecutable.

## Modelo de programación de un microprocesador de 16 bits

### 1- Modelo de memoria.

La memoria principal de este computador está formada por celdas de un byte, que constituyen la unidad básica de lectura o escritura. Cada una de estas celdas se identifica mediante una dirección. Los procesos de lectura y escritura pueden realizarse con varias celdas consecutivas simultáneamente. Para realizar cualquiera de estas operaciones el procesador deberá indicar a la memoria principal dos parámetros: la dirección de la primera celda de memoria y la longitud de la información a la que se desea acceder. El tamaño máximo de memoria lo determina la longitud de los registros de direcciones que posee el procesador, en el caso del M68000, dichos registros son de 32 bits pero por limitaciones de montaje sólo se pueden utilizar 24 líneas, es decir, la máxima longitud de la memoria principal es de  $2^{24}$  bytes (0 .. FFFFFFFF). Una palabra direccionada por el valor  $i$  se almacena con el byte más significativo en la dirección  $i$ , y el byte menos significativo en la dirección  $i+1$ . De la misma forma, una palabra larga direccionada por el valor  $i$  se almacena de modo que el byte más significativo se sitúa en la dirección  $i$ , y el menos significativo en  $i+3$ .

### 2- Modelo de los registros de datos.

Para reducir el número de accesos a memoria, (tiempo en ejecución), el procesador tiene 8 registros de datos, que se utilizan para almacenar temporalmente operandos y resultados de operaciones matemáticas y lógicas. Cada registro de datos consta de 32 bits, como el tamaño del operando puede variar desde byte (8 bits) hasta palabra larga (32 bits) en muchas instrucciones existe la opción de especificar el tamaño del dato. Este sufijo es B para indicar que los datos son del tamaño Byte, W si la operación es con palabras; y L si es con palabras largas. Los datos son almacenados comenzando desde el bit menos significativo del registro.

En las instrucciones con dos operandos el resultado de la misma se guarda en la posición especificada por el segundo operando. El primer operando se conoce como operando origen y el segundo como operando destino.

El sistema de numeración del dato se especifica al programa ensamblador mediante un símbolo que se añade al valor numérico. Si el número no va precedido por ningún símbolo, el ensamblador considera que se trata de un decimal.

Sistema de numeración	Notación
Decimal	
Binario	%
Hexadecimal	\$

### 3- Modos de direccionamiento.

Existen cuatro modos de definir operandos en el M68000, y su nomenclatura es:

- Direccionamiento inmediato: Se escribe el valor del operando precedido por #.
- Direccionamiento absoluto: Se escribe el valor de la dirección donde está guardado el operando sin prefijo.
- Direccionamiento mediante registro: Se escribe el nombre del registro de datos cuyo contenido se desea utilizar como operando.
- Direccionamiento relativo a registro: Se escribe el nombre del registro de direcciones que contiene la dirección de memoria donde está el operando. El nombre del registro se escribe entre paréntesis. Hay siete tipos distintos:
  - *Direccionamiento relativo a registro normal*: Modo clásico.

- *Direccionamiento relativo a registro con postincremento*: Trae el contenido de la posición de memoria indicada por el Registro de Dirección, y después incrementa en una cantidad unitaria de memoria dicho registro. La cantidad viene dada por el tamaño del operando.

MOVE.S (A0)+,D1

- *Direccionamiento relativo a registro con predecremento*: Decrementa en una cantidad unitaria el Registro de Direcciones y trae luego el contenido de la posición de memoria cuya dirección es el nuevo valor de dicho registro. La cantidad a decrementar viene dada por el tamaño de los operandos.

MOVE.S -(A0),D1

- *Direccionamiento relativo a registro con desplazamiento*: Hace referencia al contenido de la posición de memoria cuya dirección viene dada por la suma del valor del registro de direcciones y una cantidad fija denominada desplazamiento, que puede ser positivo o negativo. Este sistema no modifica el registro de direccionamiento y dependiendo del tamaño se incrementa N veces ( si es byte), 2N ( si es palabra) y 4n (si es palabra larga).

MOVE.S N(A0), D1

- *Direccionamiento relativo a registro con índice*: Permite utilizar desplazamientos variables, para lo que utiliza como desplazamiento el resultado de sumar un número fijo al contenido de un registro de datos ( que actúa como registro índice). Deja inalterado el Registro de Direcciones y el Registro Índice.

MOVE.S N(A0.D1),D0

- *Direccionamiento relativo a registro contador de programa con desplazamiento*: Cuando interesa hacer referencia a un operando relativo a la posición de la próxima instrucción que va a ser ejecutada. En este caso se puede utilizar un direccionamiento con desplazamiento relativo al contenido del contador de programa.

MOVE.S N(PC, D0), D1

**Formato de instrucciones:** Se denomina dirección efectiva de un operando en el M68000 a la información que identifica la situación exacta del operando. Esta información se codifica por medio de dos campos. Cada uno tiene una longitud de 3 bit. Cuando la dirección efectiva requiere incluir más información sobre la situación del operando se utiliza las palabras de ampliación. En ocasiones se utiliza la segunda palabra de ampliación para especificar otras formas de direccionamiento.

#### 4- Modelo para instrucciones condicionales.

El registro de código de condición (CCR) comprende indicadores que se almacenan en los 5 bits menos significativos de los 16 del registro de estado (SR), tienen las siguientes denominaciones : C, V, Z, N y X, ordenados de menos a más significativo, su significado es:

- C Indicador de acarreo: Indica el valor del bit de acarreo de la posición más significativa del resultado de una operación.
- V Indicador de sobrepasamiento: Indica si el resultado de una operación en complemento a dos tiene sobrepasamiento, es decir, los signos de los operandos son iguales y opuestos al del resultado.
- Z Indicador de cero: Se pone a uno cuando el resultado de una operación aritmética o lógica es 0.
- N Indicador de número negativo: Refleja el bit más significativo del resultado de una operación en complemento a dos. Indica el signo del resultado de la operación se pone a cero si es positivo y a uno si es negativo.
- X indicador extendido: Sigue las mismas leyes que el de acarreo pero sólo se ve afectado por las operaciones aritméticas y de desplazamiento, se suele utilizar para extender operaciones aritméticas de 32 a 64 bits.

#### 5- Modelo de entrada/salida.

Los computadores disponen de unos registros especializados, denominados puertos de E/S, para intercambiar información con el exterior, y son compartidos entre el procesador y los dispositivos de entrada/salida. Los puertos que se dedican a transferir información desde el entorno al procesador, reciben el nombre de puertos de entrada, lo que transfiere la información en sentido opuesto son los puertos de salida.

El M68000 considera parte del espacio de memoria como las direcciones de E/S que están conectadas a los correspondientes puertos. El mapa de memoria de un computador especifica exactamente como se distribuye el espacio de memoria entre las posiciones de memoria que van a estar disponibles para el procesador y los puertos de E/S. La implicación de conectar los puertos de E/S a la memoria es doble:

- 1- Los puertos de entrada/salida pueden escribirse y leerse utilizando todos los modos de direccionamiento de posiciones de memoria.
- 2- No existen instrucciones específicas para transferir información entre el procesador y el entorno en lenguaje ensamblador. Simplemente la información se transfiere empleando el grupo de instrucciones que se utilizan para acceder a las posiciones de memoria.

Este tipo de esquema se denomina E/S direccionado en memoria y obedece al esquema de E/S de bus único.

## **6- Modelo para la gestión de subrutinas.**

La gestión de subrutinas se realiza mediante una estructura tipo pila. Para crear una pila es necesario disponer de un espacio de memoria, donde se almacenan los datos, y de un índice que indica la posición de memoria que en ese momento es la cima de la pila.

- El espacio de memoria que se reserva a la pila se dimensiona pensando en el tamaño máximo que puede tener la pila. Dicho espacio se define a partir de dos direcciones: posición inicial (fondo de la pila) y posición máxima permitida. En el M68000 la pila crece hacia atrás por lo que la dirección de la cima será igual (si esta vacía) o menos que la del fondo. Generalmente se elige como fondo la última posición disponible de memoria.
- Puntero de pila: Indica la posición de la cima, para ello se utiliza un puntero de direcciones especial que se denomina SP.

# **TRADUCTORES, MONTADORES Y CARGADORES**

## **1- Introducción.**

Los programas escritos en lenguaje simbólico han de someterse a un proceso de transformación antes de estar listos para su ejecución. Esta transformación es automática y puede ser encargada al propio computador. Existen tres herramientas necesarias para convertir y ejecutar programas escritos en lenguaje ensamblador: programas ensamblador, montador, y cargador. El programa ensamblador se encarga de realizar el proceso de traducción de un programa fuente a un programa objeto. Es conveniente dividir los programas en pequeños módulos, que se pongan a punto por separado. Por lo que un programa ensamblador se suele diseñar de forma que no necesite tener el programa completo para proceder a su traducción, sino que sea capaz de traducir estos pequeños módulos de manera independiente. La ventaja de esta solución es que no habrá que traducir una y otra vez módulos que no se han modificado. Su principal inconveniente es que su salida no será ejecutable.

El montador se encarga principalmente de resolver las referencias cruzadas entre módulos, referencias que toman el nombre de referencias externas. Además pone en combinación los diferentes módulos que componen el programa y genera un único módulo o programa ejecutable que deposita en un fichero en el disco de sistema.

El programa cargador lleva a memoria principal los ficheros ejecutables, y una vez en memoria se encarga de lanzar su ejecución.

## **2- Programa ensamblador.**

### **• Tipos de traductores de lenguaje ensamblador.**

- Cross-Assembler: Ensamblador cruzado, se ejecuta en un computador diferente de la máquina a la que va destinado el programa objeto resultante.
- Ensamblador residente: Cuando el traductor permanece en el propio computador que ejecutará el programa. Este tipo presenta la ventaja de que el programa puede comprobarse inmediatamente sin necesidad de transportarlo de un lugar a otro. Puede presentar problemas de memoria. En máquinas pequeñas los traductores deben ser pequeños, lo que implica sencillez y con pocas ayudas al programador, por lo que el diseño del programa será más lento. Es el tipo de ensamblador indicado para el desarrollo de sistemas de control y automatismos sencillos.
- Macroensambladores: Son los traductores que permiten la manipulación de macros.
- Ensamblador de una fase: Lee una línea del programa fuente y la traduce a lenguaje máquina si se trata de una instrucción, o la ejecuta si se trata de una pseudo instrucción. Va construyendo la tabla de símbolos a medida que van apareciendo las definiciones de variables, etiquetas, etc. Esta forma de traducción obliga a definir los símbolos antes de ser empleados para que cuando aparezca una referencia a un determinado símbolo en una instrucción, se conozca la dirección de dicho símbolo y se pueda traducir la instrucción de forma correcta. Si una sentencia hace referencia a una línea posterior el ensamblador no podrá traducir la sentencia en curso ya que la etiqueta referenciada no ha sido definida. Esto se conoce como referencias adelantadas.
- Ensamblador en dos fases: En la primera fase lee el programa fuente y construye la tabla de símbolos; de esta manera, en la segunda fase vuelve a leer de nuevo el programa fuente y pueden realizar la traducción completa.

• **Estructura básica de un programa ensamblador:**

■ **Identificador de campos de una sentencia:** El análisis de campos se puede descomponer en cuatro etapas:

- Comprobar si existe campo de etiqueta. Si existe se incorporará a una tabla de símbolos que gestiona el propio ensamblador, en la que se guardan los nombres simbólicos definidos por el programador. El objetivo principal de esta etapa es asignar un valor a los nombres simbólicos.
- Comprobar si el campo mnemotécnico existe y es correcto. Para lo que el ensamblador gestiona una tabla, la tablas de códigos, donde se encuentran todos los mnemotécnicos de las instrucciones del lenguaje con sus códigos de operación asociados, así como una relación de las pseudoinstrucciones que permite utilizar con alguna indicación de su función.
- Examinar el campo de operandos y comprobar si son válidos. Etapa en que surgen dos problemas típicos, el de las referencias adelantadas y la evaluación de expresiones, cuyo objetivo es obtener un valor numérico para los operandos que vayan definidos mediante expresiones. Problema que se suele resolver limitando las posibilidades de expresiones permitidas para emplear algoritmos sencillos de evaluación de expresiones.
- Campo de comentarios, que en caso de existir será ignorado.

■ **Primera fase:** Comienza poniendo a cero el valor del contador de dirección de ensamblado. A continuación se procede a la lectura y análisis de cada línea del texto fuente. Por cada sentencia analizada se realizan las siguientes acciones:

- Añadir a la tabla de símbolos la etiqueta si existe, asociándole el valor del contador de dirección de ensamblado.
- Identificar el código de operación completo y determinar su longitud.
- Incrementar el contador de dirección de ensamblado con la longitud de la instrucción identificada.

En esta fase el programa ensamblador es capaz de detectar los siguientes errores:

- Código de operación incorrecto, si no aparece en la tabla.
- Etiqueta incorrecta, si contiene caracteres no autorizados, o es demasiado larga, o incumple alguna otra restricción impuesta a los símbolos.
- Nombre definido más de una vez, si al incluirlo en la tabla se comprueba que ya estaba en ella.

Las pseudoinstrucciones exigen un tratamiento diferente:

- Sentencias de definición de datos: La sentencia DC (definición de datos) es tratada como una instrucción máquina. La DS (espacio en memoria) no genera ningún código objeto, y en el momento que llega a ella habrá que evaluar el campo de operando, ya que este valor deberá sumarse al contador de dirección de ensamblado.
- Sentencia ORG. Su finalidad es indicar explícitamente la posición de memoria que ha de ocupar la siguiente instrucción, lo que se consigue asignando el valor al contador de dirección de ensamblado.
- Sentencia EQU. Al igual que las anteriores el valor del operando debe ser calculado en el primer paso de ensamblaje.
- Sentencia END. Da por finalizada la primera fase.

■ **Segunda fase:** Su objetivo fundamental es el de convertir definitivamente el programa fuente en los códigos binarios correspondientes. Para ellos el ensamblador vuelve a leer cada sentencia, la analiza y evalúa los operandos con ayuda de la tabla de símbolos construida en la fase anterior, y produce el código binario de acuerdo con las indicaciones contenidas en la tabla de códigos de operación.

En esta segunda fase deberá estar previsto el tratamiento de cada posible combinación de codificación fuente y formatos binarios, lo que significa que para un mnemotécnico se deberán conocer las diferentes posibilidades de modos de direccionamiento permitidos y sus efectos sobre el formato de instrucción asociado. Además el tratamiento a dar a cada operando una vez conocido el modo de direccionamiento, es decir, donde se coloca el campo de registro, el operando inmediato, el desplazamiento, etc.

En términos generales en la segunda fase para cada sentencia habrá que:

- 1- Obtener el código mnemotécnico de la instrucción.
- 2- Analizar y evaluar los operandos.
- 3- Localizar el código de operación binario en la tabla.
- 4- Determinar los modos de direccionamiento de los operandos.
- 5- Realizar el tratamiento correspondiente.
- 6- Incrementar el contador de dirección de ensamblado con el tamaño básico más el de las palabras de ampliación.

■ **Consideraciones de diseño.** Para diseñar un programa ensamblador rápido y eficaz se deben de cumplir las siguientes condiciones:

- Limitar el análisis a una sola vez.



- Gestionar eficientemente la tabla de símbolos.
- Tener un buen tratamiento de errores.

### **3- Macroensamblador.**

Si las definiciones de macros se codifican antes de cualquier llamada, la expansión podrá realizarse en una sola fase.

- **Preensamblaje. Traducción en tres fases.**

Se llama ensamblado preensamblado al proceso de expansión de los macros cuando tiene lugar en forma independiente, antes del ensamblaje del nuevo texto expandido.

El macro ensamblador examina las líneas una a una, cuando encuentra una pseudoinstrucción MACDEF registra toda la información, hasta que encuentra MACEND, en memoria para su uso posterior, y anota en una tabla de macros el nombre de la macro y un puntero a donde se almacena su definición.

Cuando se localiza una llamada a una macro se mira en la tabla de macros si existe, en caso afirmativo se compara la sentencia completa con el prototipo correspondiente creando una tabla de variables temporal, en la que se escriben los nombres de los argumentos simbólicos de la macro y los valores asignados en esa llamada. A continuación se copia en el texto expandido la definición íntegra de la macro ( hasta MACEND o MACEEXIT), sustituyendo los argumentos simbólicos (que comienzan por &) por los valores asignados en esta tabla.

Dada la posible recursividad del proceso de expansión, la tabla de variables tendrá estructura de pila, y en ella se irán acumulando las posibles llamadas encadenadas que se produzcan.

- **Expansión en la primera fase. Traducción en dos fases.**

En la expansión de macros se va obteniendo el resultado línea por línea, por lo que no hay inconveniente en que cada sentencia del texto expandido sufra el análisis correspondiente a la primera fase de ensamblaje. La obtención del programa objeto final se puede realizar en dos fases. Tiene como inconveniente el exigir mayor espacio en memoria ya que hay que mantener más tablas de información simultáneamente.

### **4- Programa montador.**

#### **Combinación de módulos de programa.**

La misión principal del programa montador es combinar uno o más módulos objeto en un único módulo ejecutable. Las técnicas más frecuentemente empleadas para reunir las diferentes partes que componen un programa son:

- **Módulo fuente.** Varios módulos fuente pueden combinarse reuniendo los programas fuente correspondientes en un texto único, que ensamblará obteniendo un programa objeto listo para ser ejecutado. Los programadores deberán ponerse de acuerdo previamente para evitar repetir nombres simbólicos; otro inconveniente es el empleo de subprogramas biblioteca, cada vez que se utilicen cualquiera de ellos se volverá a efectuar su traducción por la correspondiente pérdida de tiempo.

- **Módulos objeto.** En la combinación anterior el ensamblador sitúa un módulo a continuación de otro, salvo que voluntariamente se especifique lo contrario mediante la sentencia ORG. Al combinar los módulos objeto, será el usuario quien deberá asignar una zona a cada uno, asegurándose de que no se produzcan solapes entre zonas, e informar al ensamblador mediante las sentencias ORG oportunas, que ahora son imprescindibles.

Durante el desarrollo y puesta a punto del programa el trabajo de ensamblaje puede ser ,más reducido, ya que si las modificaciones se hacen sobre un único módulo sólo habrá que reensamblar éste. El principal inconveniente de esta técnica es que en la memoria del computador pueden quedar bastantes huecos entre módulos.

#### **El problema de la reubicación.**

Las dos formas anteriores resuelven el problema de la creación de programas complejos, aprovechando las ventajas de la programación modular. No obstante ninguna de las dos resulta cómoda para el programador, ya que la primera le obliga a reensamblar todos los módulos cada vez y la segunda a vigilar personalmente que no se solapen los módulos objeto, además de desaprovechar parte de la memoria.

Lo ideal sería poder combinar módulos objeto unos detrás de otros sin tener que preocuparse de su colocación en memoria. Esto se puede conseguir si se dispone de un mecanismo que permita modificar un módulo objeto, de forma que pueda colocarse en otra zona de memoria distinta a aquella para la que fue preparado, sin tener que reensamblarlo. Esta operación se denomina reubicación. La forma habitual de hacerlo consiste en asignarles un origen común a todos ellos. Para que un programa reubicado funcione correctamente será preciso modificar los códigos de las instrucciones de direccionamiento absoluto.

**Ensamblado para reubicación.** El problema de la reubicación puede resolverse automáticamente si en el programa objeto aparecen debidamente marcadas todas las instrucciones que deben ser modificadas. Bastará con construir durante el ensamblado una tabla de direcciones absolutas que contenga todas las posiciones de memoria donde se utilizan direcciones absolutas, anotando por cada una de ellas dicha dirección. Una vez determinado el valor T de la traslación, el trabajo del responsable de reubicar el programa ( a veces el programa montador y otras el cargador) se reducirá a incrementar en T el valor de todas las posiciones indicadas en esta tabla.

Si un programa todos los direccionamientos son relativos, dicho programa se dice que es autoreubicable, ya que podrá funcionar en cualquier zona de memoria sin necesidad de modificar ningún código de instrucción.

#### **El problema de las referencias externas.**

Se produce cuando un módulo necesita utilizar datos o zonas de código de otros módulos. Al ensamblarlo no se podrá completar la tabla de nombres simbólicos, lo que en principio se tomaría como un error de programación. Para solucionarlo se emplea dos subinstrucciones DEF y EXT. La sentencia EXT permite que el ensamblador reconozca una referencia como externa. El programa montador deberá solucionar estas referencias externas, pero en ensamblador debe dejarle el trabajo preparado. Para ello ha de producir dos tablas adicionales, la tabla de símbolos globales que contendrá los nombres simbólicos que el módulo exporta a los demás, y la tabla de símbolos externos, formada por los nombres simbólicos que necesita importar de los demás.

Para que se pueda realizar correctamente el enlace entre módulos, tanto los nombres simbólicos globales como los externos deberán aparecer de alguna forma en el programa objeto. En esto presentan una diferencia fundamental con los demás nombres simbólicos elegidos por el programador, que desaparecen totalmente una vez realizado el ensamblaje, sin quedar reflejados de ninguna manera en el código generado.

Los símbolos externos sólo podrán utilizarse como referencias puras (no en expresiones) con lo que bastará que la tabla de símbolos externos tenga una anotación por cada aparición de uno de estos nombres simbólicos, Cada entrada de esta tabla deberá contener:

- El nombre simbólico.
- La dirección relativa que aparece dentro del módulo
- Su tipo: dirección o parámetro ( en este caso su longitud)

En el caso de la tabla de símbolos globales será suficiente con los siguientes datos por cada símbolo global declarado:

- El nombre simbólico.
- Su tipo: dirección o parámetro ( en este caso su longitud)
- Valor asignado al valor o parámetro.

#### **Funcionamiento del programa montador**

Además de convertir uno o varios módulos objeto en un único módulo ejecutable, generalmente se le asignan las funciones de reubicar los módulos a su zona de ejecución y de resolver las referencias externas de los módulos. Para realizar su trabajo requiere la colaboración del ensamblador, ya que necesita que éste le elabore por cada módulo la siguiente información:

- Indicación de si es un módulo absoluto o reubicable.
- Si es absoluto su dirección de carga.
- Tabla de direcciones absolutas para la reubicación.
- Tabla de símbolos globales.
- Tabla de símbolos externos.

Además el usuario le debe informar de:

- Los nombres de todos los módulos a montar.
- Las posiciones absolutas donde colocar las partes del programa que son reubicables.

Una vez conocida la dirección inicial se procede a la reubicación, lo que se consigue aumentando en el valor de la dirección inicial, todas las direcciones absolutas marcadas en su tabla de direcciones absolutas y todas las direcciones en la tablas de símbolos globales. Una vez reubicados todos los módulos, ya se tienen las direcciones absolutas de todos los nombres simbólicos globales, con lo que se puede solucionar las referencias externas. Para ello se construye una tabla general de símbolos globales, obtenida uniendo todas las tablas de los diferentes módulos. Una vez construida se pueden resolver las referencias externas de cada módulo. Finalmente, el montador debe copiar todos los módulos en un sólo bloque ejecutable, que pueda ser manejado por el cargador, para proceder a su ejecución.

El bloque ejecutable total se divide en pequeños bloques de direcciones sucesivas, añadiendo a cada uno de ellos una cabecera con su dirección de carga y su longitud. Adicionalmente se pueden añadir a cada bloque un código de detección de error.

#### **Utilización de las bibliotecas de subrutinas.**

Las bibliotecas o librerías de subrutinas están formadas por una serie de módulos ensamblados de forma reubicable, lo que le asigna más trabajo al montador. Existen dos filosofías de trabajo: el montador busca los módulos externos en todos los ficheros del sistema que sean del tipo librería, o bien limita la búsqueda a las librerías explícitamente especificadas en la orden de montaje.

### ***8- Programa cargador.***

Tiene como misión leer un fichero ejecutable e un medio de almacenamiento externo y traspasarlo a la memoria principal, para proceder a su ejecución. Su estructura es muy simple debe de tener un bucle que se encargue de leer y almacenar las palabras de un bloque, y de ir calculando su código detector de errores. Este se encontrará anidado dentro de otro bucle general que se repetirá tantas veces como bloques tenga el fichero.

En algunas aplicaciones es importante poder cargar los programas en distintas zonas de memoria, dependiendo de ciertas condiciones de ejecución. Para ello el cargador debe ser de tipo reubicador, entonces, debe ser capaz de posicionar el programa en la zona de memoria deseada, para lo que deberá disponer de una tabla general de direcciones absolutas que permita realizar la reubicación.

#### **Arranque del computador**

Una técnica simple que se utiliza en muchos computadores consiste en implantar una parte muy pequeña de la memoria principal en forma de memoria de sólo lectura. Esta ROM contiene una cuantas instrucciones conocidas como cargador inicial (bootstrap) que son suficientes para transferir otro cargador más completo desde una posición fija de la memoria de un dispositivo de almacenamiento masivo del sistema a una posición fija de la memoria principal. Una vez que el cargador está en memoria principal, el sistema podrá cargar automáticamente en dicha memoria cuantos programas necesite para funcionar con normalidad.