UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE CIENCIAS FISICO-MATEMATICAS

La Computación en la Telefonía

TEMA:

QUE PARA OBTENER EL TITULO DE:

Licenciado en Ciencias Computacionales

PRESENTA:

Alfredo Cortés Aguirre

SAN NICOLAS DE LOS GARZA, N. L.

MARZO DE 1985

20 mar 85

19 horas

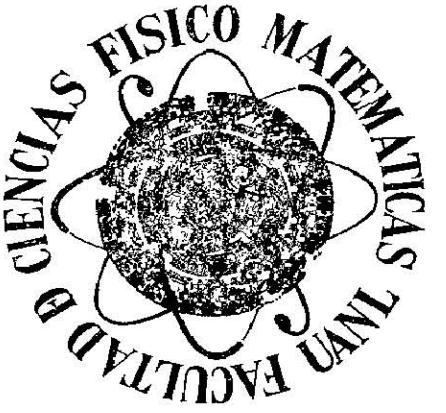
Alfredo Cortes Aguirre

tel. Dom. 70-57-12

tel. ofna. 45-18-66

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE CIENCIAS FISICO-MATEMATICAS



La Computación en la Telefonía

T E M A :

QUE PARA OBTENER EL TITULO DE:

Licenciado en Ciencias Computacionales

P R E S E N T A :

Alfredo Cortés Aguirre

Gracias por todo el apoyo recibido durante el desarrollo del proceso
¡ Gracias !

SAN NICOLAS DE LOS GARZA, N. L. MARZO DE 1985

LA COMPUTACION EN LA TELEFONIA

A:

GRECIA MARIA,

FATIMA

Y

MARTHA DELIA

GRACIAS A TODOS LOS QUE ME
YUDARON EN EL DESARROLLO DE
ESTE TRABAJO.

I N D I C E

I N T R O D U C C I O N

A través de la Historia de las culturas de los diferentes países, la comunicación ha sido uno de los factores primordiales para su desarrollo; así también como la evolución de las matemáticas.

En este tema se tratará un medio de la comunicación: la Telefonía; y un instrumento de las Matemáticas que son las computadoras; que aparentemente a través de la Historia no tenían ninguna relación pero que analizadas cuidadosamente una y otra se apoyaron mutuamente hasta su evolución actual.

HISTORIA DE LA TELEFONIA

Primeros Medios de Comunicación a Distancia.

Los primeros medios de comunicación que la humanidad utilizó, fueron los mensajeros que transmitían en forma verbal o escrita el mensaje. Años más tarde se valieron de estafetas humanas, relevos, que llevaban el mensaje a grandes distancias; después emplearon animales rápidos, como caballos y palomas mensajeras.

También se utilizaron otros medios de comunicación en forma de señales ópticas y acústicas, como hogeras, atalayas, banderas tambores, espejos, etc. Algunos de éstos sistemas se encuentran en uso aún hoy en día, especialmente en el ejército y la marina.

En 1792 el Ingeniero francés Claudio Chappe inventó, junto con su hermano Ignacio, el telégrafo óptico que representó un gran progreso en las comunicaciones. Por medio de un poste provisto en su parte superior de un travesaño compuesto de dos brazos manejables por medio de cables, logró enviar a considerables distancias mensajes en clave. Se colocaba el aparato sobre sitios elevados, y mediante catalejos podían observarse las señales desde distancias hasta de 60 Km. El telégrafo óptico adolecía de evidentes deficiencias, solo servía para distancias muy limitadas y su utilidad se nulificaba por completo cuando había niebla.

Solamente después de 1800 cuando uno de los fundadores de la ciencia eléctrica, el físico italiano Alejandro Volta, dió a conocer la pila eléctrica fué posible por vez primera hacer experimentos de comunicación empleando la corriente eléctrica. El 20 de julio de 1820 Juan Cristián Oerested, profesor de la Universidad de Copenhague en Dinamarca, descubrió la estrecha relación que existe

entre el magnetismo y la eléctricidad. Poco tiempo después el físico y matemático francés Andrés María Ampere estudiaba este fenómeno y establecía las leyes sobre las que descansa el electromagnetismo.

Arago y Davy descubrieron que haciendo pasar una corriente eléctrica por un hilo aislado enrrollado sobre una pieza de hierro se transformaba en un imán, y en 1825 el físico inglés Guillermo Sturgeon construía el primer electroimán en todo idéntico a los que usamos actualmente.

Faraday y Henry descubrieron independientemente uno de otro en 1835 la reciprovidad de las leyes del electromagnetismo. Faraday demostró que si un circuito eléctrico abarcaba un campo magnético y se modificaba la fuerza del campo, se generaba una corriente en el circuito eléctrico.

Invención del Telégrafo por Morse.

Al enterarse Samuel Morse de los experimentos de Miguel Faraday sobre el electromagnetismo proyectó la construcción de un instrumento telegráfico registrador y estableció los principios relativos a su clave de puntos, guiones e intervalos, fundada en la duración o la ausencia de puntos eléctricos.

En 1835, Morse construyó un modelo experimental en el que la acción mecánica de un electroimán hacía funcionar una palanca que sostenía un lápiz. El paso de impulsos eléctricos a través del electroimán hacía que la cinta se moviese sobre la superficie de una cinta de papel. A medida que ésta avanzaba sobre un cilindro situado debajo del lápiz se iba trazando una línea que incorporaba la clave de Morse. Fué en 1837 cuando Morse solicitó la patente de

la invención del telégrafo electromagnético.

La línea telegráfica de los primeros experimentos llegaba hasta unos 30 Km y se pudieron prolongar cuando Morse inventó un repetidor que permitía llegar las señales a grandes distancias.

Las comunicaciones a base de impulsos eléctricos se iniciaron en 1844 con la operación de la línea telegráfica entre las ciudades de Washington y Baltimore sobre la cual Morse transmitió su frase célebre: "¡ Que ha forjado Dios ."•

Invención del Teléfono por Bell.

El primer descubrimiento que sugirió la posibilidad práctica de transmitir la voz humana, se debe al americano Carlos G. Page de Salem. Page y Henry comprobaron en 1837 que haciendo pasar una corriente alterna a través de un selenoide provisto de un núcleo de hierro, éste emitía sonidos debidos a la alteración molecular causada por el cambio de sus condiciones magnéticas: es decir, el campo magnético alterno producido por la corriente eléctrica hace vibrar las diferentes partículas que se atraen y se repelen alternativamente dentro de los límites de la elásticidad del hierro. A este fenómeno se le conoce hoy en día como "Efecto de Page".

La primera idea sobre la transmisión de las palabras se debe al telegrafista militar Carlos Bourseull originario de Bruselas que en 1854 escribía: "Hablando delante de una membrana que establezca e interrumpa sucesivamente la corriente de una pila, y enviándola a una línea terminada en un receptor formado por un electroimán, éste podrá atraer y soltar una placa o armadura móvil. Es indudable que de esta suerte se llegara en un porvenir más o menos

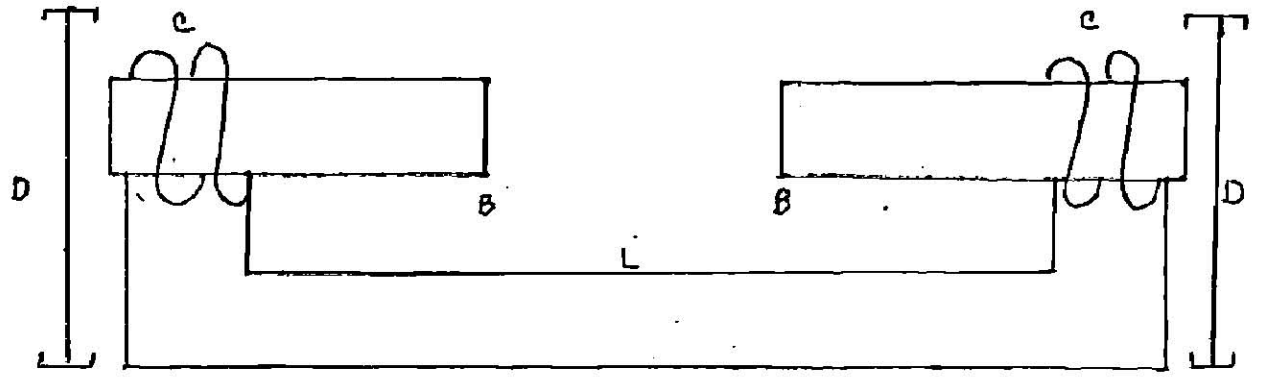
próximo a transmitir a distancia por medio de la electricidad, la palabra. Las sílabas se reproducirán exctamente por la sola vibración de los medios interpuestos. Reproduciendo estas vibraciones se reproducirán exctamente las sílabas".

En 1876 los profesores Alejandro Graham Bell y Eliseo Gray solicitaron el mismo día 14 de febrero la patente de invención del teléfono, y los tribunales de justicia de los Estados Unidos después de largos debates concedieron la patente a Bell natural de Edimburgo, Escocia, que había emigrado a América en 1870 donde se naturalizó.

El 10 de marzo de 1876, hallándose Bell perseverando en sus investigaciones, auxiliado por Watson, en el último piso de su casa de Boston el éxito coronó sus trabajos. Watson que se hallaba en otra pieza distinta oyó las primeras palabras producidas por el teléfono y pronunciadas por Bell: " Mr. Watson, venga lo necesito ".

En la figura podemos observar el teléfono de Bell formado por una barra imanada B que llevaba en el extremo más próximo a la armadura una bobina C de hilo fino de cobre. La armadura estaba formada por un delgado D diafragma de hierro montado de manera que pudiera vibrar libremente frente al polo del imán. Este aparato hacía las veces de transmisor y receptor, uniéndose ambos por un conductor eléctrico L.

En la primitiva comunicación telefónica no se utilizaba ninguna pila eléctrica y el transmisor era idéntico al receptor.



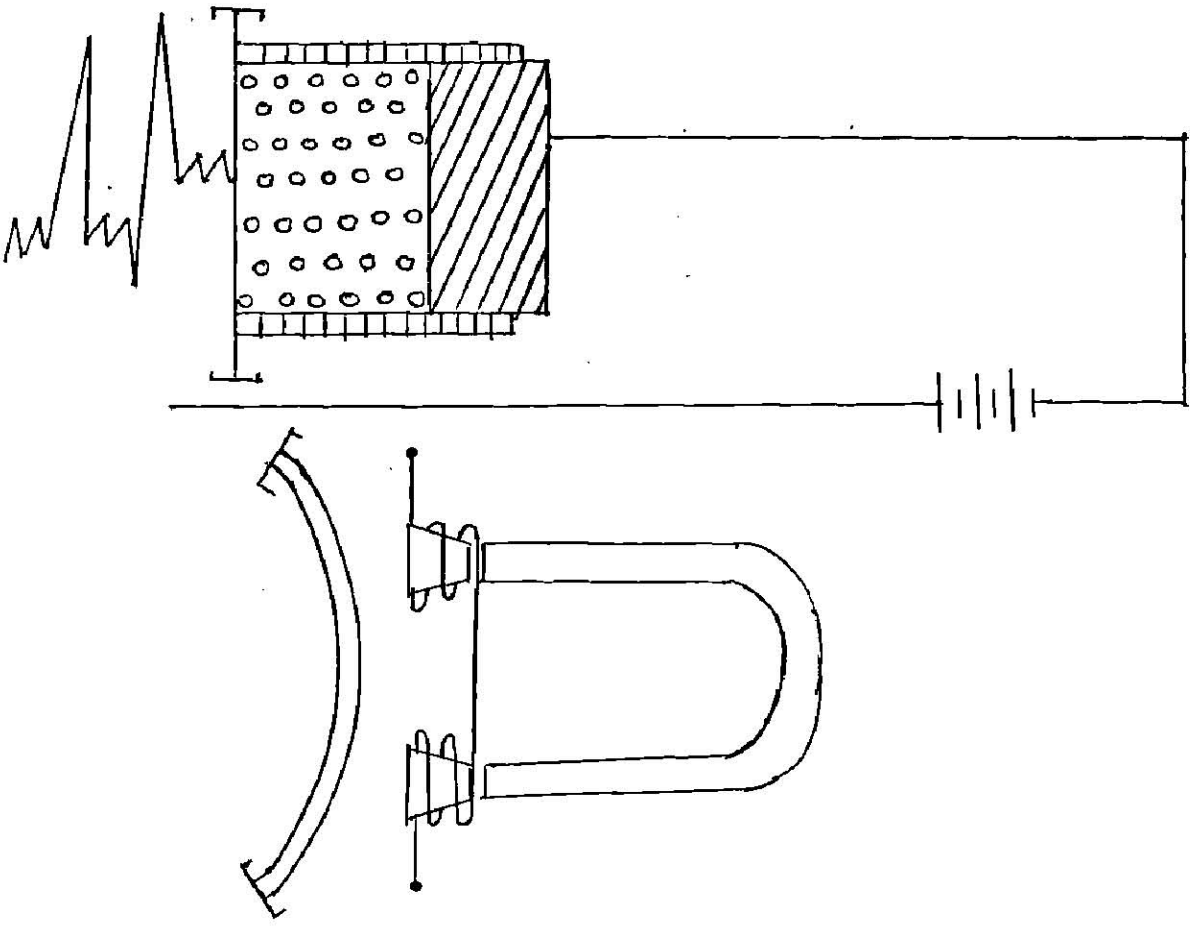
Transmisor
Receptor

Receptor
Transmisor

Cuando se habla delante de la membrana D del transmisor, las vibraciones hacen variar el flujo magnético cortado y engendra por consiguiente corrientes de inducción en la bobina C que circulan por la línea L y atraviesan la bobina C del receptor, modificando su magnetismo y haciendo vibrar su membrana en concordancia con las vibraciones de la membrana del transmisor con lo que dá lugar a la reproducción de la voz humana.

Los teléfonos magnéticos dieron resultados satisfactorios cuando la distancia entre el receptor y el transmisor era corta - pero no sucedía lo mismo cuando aquélla era muy grande. El primer transmisor que se empleó para solucionar este inconveniente fué el de carbón, ideado por Tomás Alba Edison. Después de Edison no tardó el Profr. Hughs en inventar en 1876 su transmisor utilizando por primera vez el término micrófono.

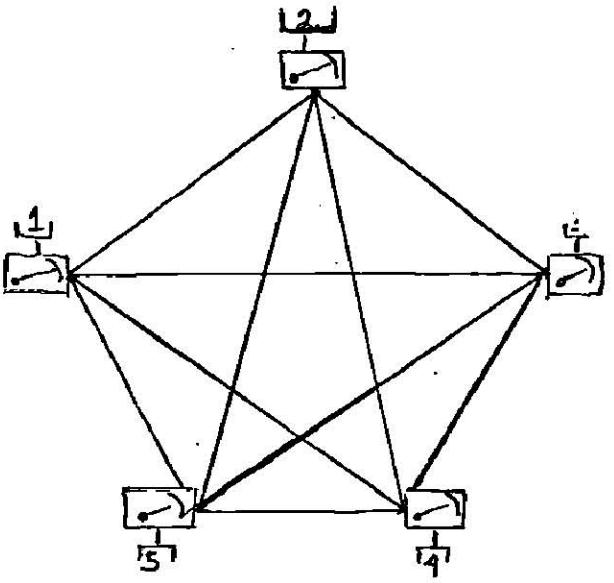
El micrófono de carbón de Hughs fué la base de las transmisiones de los teléfonos con pila como se indica en la figura.



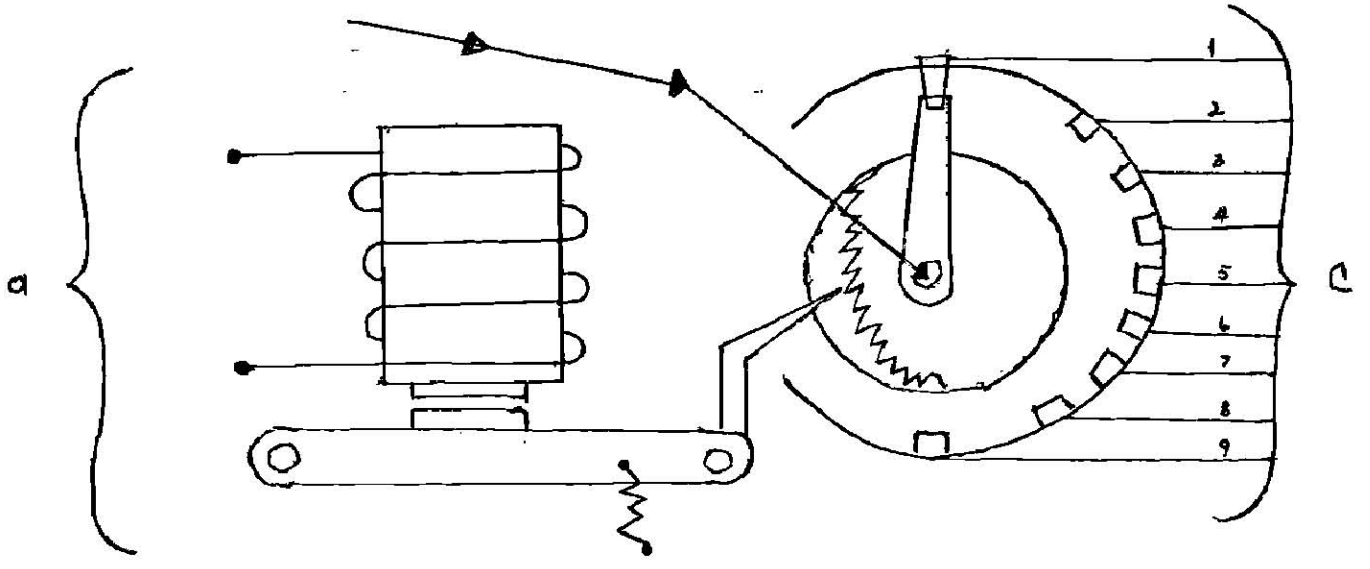
Al dispositivo que convierte los impulsos eléctricos en ondas sonoras, se le dá en telefonía el nombre de audífonos y trabajan mediante el sistema de agrandar o disminuir el campo magnético del electroimán haciendo que el diafragma se contraiga, reproduciendo los sonidos o palabras.

La primera central telefónica se inauguró en New Haven, Connecticut, el 23 de enero de 1878 siguiendo después las de Bridge Port, New York, Filadelfia, Chicago, etc.

La comunicación automática nació en Estados Unidos y fué ideada en 1891 por Almond B. Strowgr de Kansas City. La primera central automática funcionó en Laporte, Indiana en 1892.



Como se puede apreciar en la figura anterior la idea de centralizar la red telefónica nació de la necesidad cuando se tuvo que aumentar el número de aparatos telefónicos. Posteriormente al crecer la demanda de aparatos telefónicos y de ser un sistema que se conectaba manualmente nació la idea de automatizar el servicio telefónico.



En la figura se observa el primer tipo de selector automático que fué diseñado para las primeras centrales automáticas, pues

dependiendo de la cantidad de impulsos enviados a través de un disco en el aparato telefónico, este accionaba la magneta que a su vez halaba la palanca que tenía un brazo que al ser accionado movía la rueda dentada tantos engranes como impulsos recibía la magneta, de tal manera que seleccionaba uno de los números del múltiple C. Años después se desarrollaron otros selectores que eran accionados con más relevadores con más engranes, hasta que en los años 30s se desarrolló un selector de coordenadas que también era electromecánico pero ya sin utilizar engranes.

Fué de la cantidad de llamadas que se manejaban que se desarrolló un "registro" para poder manejar mejor las llamadas. Este "registro" recibía la información, la almacenaba, la codificaba, la analizaba, la procesaba y nos conectaba con el número al que se deseaba llamar. Este "registro" era el procesador automático de la telefonía que también era electromecánico.

Fué hasta los años 60s con la invención del transistor como se empezaron a desarrollar los primeros procesadores electromecánicos y en 1968, que en Estokolmo, Suecia, se inauguró la primera central totalmente electrónica con una tecnología en sus componentes que hoy en día nos parece anticuada.

En 1975 fué cuando se empezaron a usar los circuitos de alta integración en algunos sistemas telefónicos y hoy en día se emplean cada vez más en muchos sistemas gracias a la alta integración, al bajo costo y la variedad de los mismos circuitos.

HISTORIA DE LA COMPUTACION

Indudablemente que el dispositivo más antiguo de computación son, sin lugar a dudas los cinco dedos de cada mano y este es aún el instrumento más preferido de todo niño que aprende a contar. Puesto que se tienen diez dedos discretos (digitos) para la cuenta. Tanto la computación digital como el sistema decimal, han gozado de enorme popularidad en la Historia.

El ábaco es uno de los primeros inventos que se utilizaron en China hace más de 2 000 años, y aún hoy en día lo usan muchos comerciantes chinos y japoneses.

La verdadera iniciación de las computadoras modernas se remonta al siglo XVII de donde parte nuestra "era moderna". Fueron intelectuales como Descartes, Pascal, Leibnitz y Napier los que formaron un nuevo comienzo en la Filosofía, la Ciencia y las Matemáticas. En Matemáticas particularmente se hicieron progresos tan tremendos y los cálculos concurrentes fueron tan laboriosos que se volvió urgente la necesidad de máquinas computadoras más sofisticadas.

El desarrollo de los logaritmos por John Napier en 1614 y su conversión a la base 10 por Henry Griggs en 1615, estimuló la invención de varios dispositivos que sustituían la adición de los logaritmos en lugar de la multiplicación. Uno de estos dispositivos inventado por John Napier en 1617 fué uno mecánico de varillas numeradas que podían multiplicar. Estos se conocieron más tarde como los "huesos de Napier". Una regla de cálculo sin partes movibles basada en los logaritmos de Napier, fué inventada por Edmund Gunter. Esta fué mejorada por la introducción de una escala deslizante por William Oughtred. Esta fué la precursora de la regla de cálculo mo-

derna.

Quizá la mayor significación en la evolución de las calculadoras mecánicas fué la introducción en 1642 de las "ruedas dentadas" (engranes) por Blaisie Pascal. Aunque su uso se limitaba a la adición y sustracción la "rueda dentada" para contar se usa aún en máquinas sumadoras. El filósofo y matemático alemán Barón Von Leibnitz incorporó la multiplicación a la máquina de Pascal en 1671, siguiendo la adición repetida de un número. Fué hasta 1820 que Tomás de Colmar mejoró la calculadora de Pascal para hacerla práctica en la multiplicación.

La primera máquina digital programada, probablemente se remonta al telar de tarjeta perforada inventada en 1801 por Jacquard operando en forma similar a una pianola, la máquina Jacquard logró un control de proceso digital automático de telas con figuras tejidas, por medio de un telar controlado por tarjetas perforadas. Charles Babbage, aplicó la idea de la tarjeta perforada para programar su "máquina analítica" (en 1833) que contenía todos los conceptos de una verdadera computadora automática que sin embargo, nunca fué completada debido a la tecnología insuficientemente desarrollada de la época. No fué sino hasta 1886 que Hollerith en la oficina de censos de los Estados Unidos, desarrolló una máquina eficaz para tarjetas perforadas, que sirvió para clasificar y tabular datos de censos.

El crédito para la primera computadora digital realmente automática en gran escala, es para el profesor Howward Aiken de la Universidad de Hardvard. Usando muchas de las ideas germinales de Babbage, Jacquard y Hollerith, el profesor Aiken, operando con la

International Businness Machines Corporation (IBM), desarrolló de 1937 a 1944 el calculador automático de secuencia controlada, que se conoció como Mark I. El prototipo de todas las computadoras digitales automáticas era esencialmente electromecánico en su operación y contenía un crecido número de interruptores, relevadores, ruedas, contadores, contactos de levas, etc. haciendo un total de más de 760 000 partes. Este gran número de partes explica el tiempo que transcurrió para la manufactura de computadoras. Antes de los 30s los dispositivos mecánicos y eléctricos no eran suficientemente eficaces para que una máquina grande trabajara correctamente.

La calculadora Mark I Hardvard - IBM tenía todos los componentes esenciales y funcionales de una computadora digital automática: entrada, memoria, unidad aritmética (de proceso), control y salida, excepto que su acción de computadora propia (aritmética) no está separada, como en los tipos ulteriores de computadoras, sino que estaba unida a las operaciones de memoria. La entrada de la máquina, consiste de veintitrés números decimales digitales e instrucciones de operación, es alimentada por tarjetas perforadas IBM, cinta perforada o interruptores de cuadrante de ajuste manual. Dependiendo de las instrucciones codificadas, la máquina puede detectar automáticamente cualquier secuencia de operaciones que se desee, por ejemplo suma, resta, multiplicación, división y transferencia o limpieza de números, así como cálculo de logaritmos exponenciales, funciones senoidales, etc. Sin embargo, de acuerdo con las normas actuales, la computadora Mark I es lenta. Esto se compara con la fantástica velocidad de las computadoras electrónicas recientes, que pueden efectuar las operaciones metamáticas en unas cuantas mi-

llonésimas de segundo (microsegundos) o menos.

Aunque se construyeron computadoras electrónicas más avanzadas, basadas en la Mark I, en los 40s en Hardvard, los esfuerzos industriales se dirigieron hacia las computadoras digitales electrónicas, más rápidas y más eficaces. La primera llamada ENIAC (Electronic Numeral Integrator an Calculator) construída en 1942 en la escuela de Ingeniería Electrónica de Moor, en la Universidad de Pensylvania.

La ENIAC de la Moor, contiene 18 000 tubos electrónicos. Su aparición hizo anticuadas todas las computadoras de relevadores debido a su capacidad para efectuar 5 000 adiciones en un segundo, en comparación con la velocidad máxima de 5 a 10 adiciones por segundo de una computadora de relevadores.

El desarrollo iniciado por la ENIAC en 1945 fué seguido de un verdadero alud de computadoras electrónicas en los años subsecuentes y que fueron nombradas pintorezcamente como: EDVAC, ORDVAC, BIZMAC, SEAC, RAYDAC, UNIVAC, NORC, LARC, RAMAC, IBM 701-705, MUSE, MOBIDIC, STRETCH, etc. Estas computadoras han sido cada vez más sofisticadas y su precisión, eficacia, velocidad y capacidad de almacenamiento de memoria, han sido mejoradas.

FUSION DE LA TELEFONIA Y LA COMPUTACION

Fué después de la invención del teléfono cuando se inició el diseño de la automatización de las diferentes centrales telefónicas que podemos observar como los primeros diseñadores de sistemas telefónicos los cuales se vieron en la necesidad de recurrir a la tecnología existente en su tiempo, y fué de ésta manera como se apoyaron en las máquinas sumadoras.

La idea consistía en almacenar el número deseado, codificarlo y después convertirlo en impulsos eléctricos para seleccionar mediante engranes y relevadores el número deseado. Cuando se aumentaba el número de aparatos telefónicos se aumentaba el número de selectores en serie pues se tienen que hacer varias "selecciones" en una sola llamada.

Posterior a esto, la telefonía tuvo un gran desarrollo mientras que las computadoras solo se redujeron a simples sumadoras que algunas veces también multiplicaban.

Para 1940, antes de que se desarrollara la primera computadora, la Mark I, la telefonía ya contaba con centrales que tenían relevadores más eficaces y de múltiples aplicaciones, además de selectores de coordenadas que eran mucho mejores que las anteriores pues ya no contaban con los engranes que las hacían lentas.

De esta manera fué como la telefonía sirvió de ejemplo para las primeras computadoras basadas en relevadores. La computación después, tuvo un gran desarrollo, de tal manera que fué hasta los años 60s en que se tuvo la idea de cambiar los registros de la telefonía (procesadores) por una computadora que hacía las mismas funciones. Hasta entonces fué que se concibe la idea de la fusión de

la telefonía y la computación, pero que en realidad han sido de cierta manera dos ciencias que han estado en alguna forma ligadas una a otra.

TRANSMISION DIGITAL

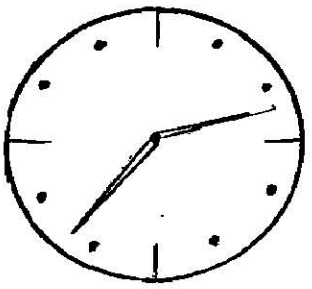
Señales Analógicas y Digitales.

Los términos: analógica y digital son calificativos que se le pueden aplicar a una información o una señal alectrónica dependiendo de los valores que ésta pueda tomar; así el término digital proviene de la palabra dígito lo que a su vez significa número; esto es, cuando una información solo puede expresarse mediante un grupo fijo de cifras, o sea de números, esta información puede considerarse como digital.

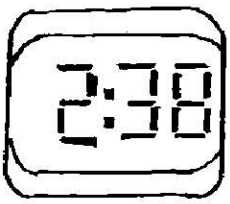
Un ejemplo de información digital es el indicador de un elevador, el cual solo nos dice si el ascensor se encuentra en el piso, 1, 2, ... etc., pero no hay ningún foco que nos indique que está en el piso 2.37 por ejemplo.

Por otro lado las señales analógicas son aquellas que pueden tomar cualquier valor entero o fraccionado dentro de un rango; un ejemplo sería el indicador de velocidad de un automóvil, en el cual la aguja puede girar libremente y ocupar cualquier posición alrededor de la carátula, indicándonos así la velocidad aproximada; pero no nos dá una lectura precisa en "dígitos" o números enteros.

Otro ejemplo muy común son los relojes, a los cuales cuando son de manecillas se les llama analógicos, mientras que si nos dan la hora en cifras exactas se les llama digitales, y así podríamos enumerar una gran cantidad de indicadores o de señales a los cuales aplicarles estos dos calificativos: Analógico o Digital.



RELOJ
ANALOGICO



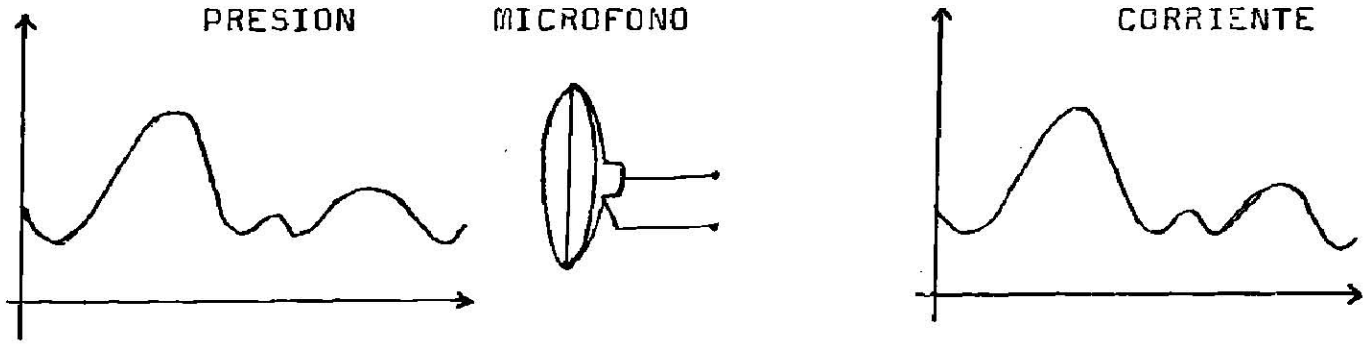
RELOJ
DIGITAL

INFORMACION ANALOGICA Y DIGITAL

Naturaleza Analógica de la Voz.

Refiriéndonos ahora a la telefonía y recordando ahora que su objetivo primario y principal es la transmisión de la voz a distancia, analicemos un poco la forma tradicional en que esta transmisión se ha efectuado.

El sonido en general es en esencia una serie de cambios de presión en el aire que circunda al emisor de dicho sonido; así la voz que es una clase de sonido, es emitida al vibrar las cuerdas y cavidades bucales de una persona, y puede tomar cualquier volumen y cualquier frecuencia dentro de un rango determinado. En la figura adjunta se puede ver a la izquierda una representación gráfica de esto, en ella se dibujaron los valores que la presión en el aire va



SEÑALES ANALOGICAS DE VOZ (SONIDO Y SEÑAL ELECTRICA)

tomando en cada instante de tiempo; como se ve esta señal es analógica por naturaleza.

Para transmitir la voz, lo primero que se hizo fué convertir esta señal a alguna otra forma que pudiera enviarse a distancia con mayor facilidad que la voz misma, esto lo logró por primera vez Alexander Graham Bell al inventar el micrófono que convierte los cambios de presión en cambios en la intensidad de corriente que lo atraviesa, pero como vemos en la figura esta señal sigue siendo analógica.

Para transmitir esta señal normalmente se utiliza un par de alambres metálicos, e incluso desde hace algunos años y conforme los adelantos en los dispositivos electrónicos lo hicieron posible, se han empezado a usar semiconductores para transmitir estas señales; pero de todas formas, no importa si el equipo sea electromecánico o electrónico, la comunicación de voz seguia siendo en forma analógica.

Transmisión Digital de la Voz.

El término "Telefonía Digital" está relacionado con la forma en que se transmite la voz en un sistema telefónico, pero ¿Cómo puede ser esto posible si ya hemos visto que tanto la voz como la señal que el micrófono produce son analógicas?. La solución es muy sencilla, solo basta medir a intervalos regulares el nivel de la señal eléctrica que el micrófono produce, asignarle a cada medida un valor numérico exacto, y enviar este número hacia el otro extremo de la línea, en donde se generará una señal idéntica a la que se usó para tomar mediciones.

Ahora las Administraciones están reconociendo el potencial

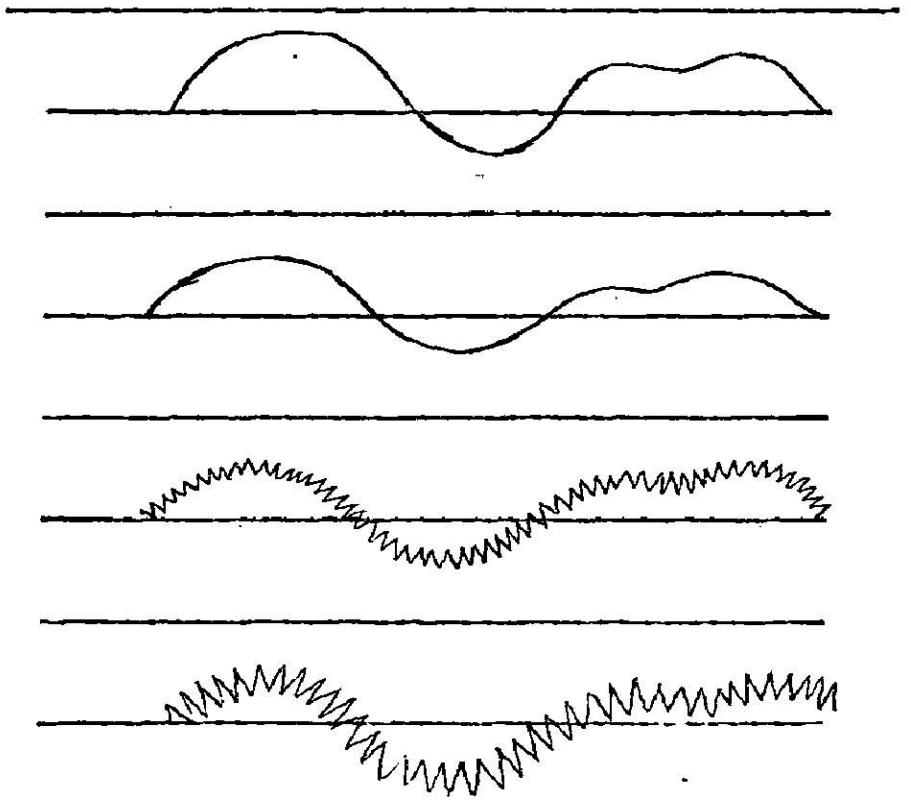
de las redes totalmente digitales, y el inconveniente que tenían las redes independientes para servicios diferentes como (Télex, cablevisión, faccimil, teleinformática y telefonía) y comienzan a pensar en la IDN para la digitalización completa, incluyendo el aparato de abonado y la integración completa de los servicios modernos de voz y datos.

IDN (del inglés Integrated Digital Network).

Ventajas de la Transmision Digital.

¿Por qué se prefieren frecuentemente las señales digitales para el envío de información, aún para aquellas que son inherentemente analógicas como voz o imágenes?

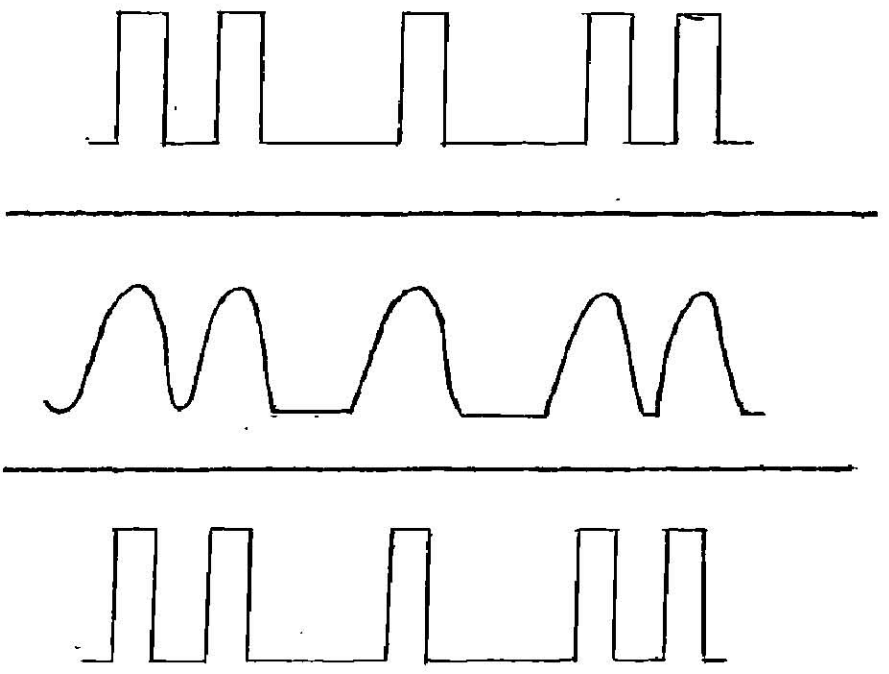
Una razón importante es que la señal digital tiene mucho menos ruido y distorsión y, por tanto, mejor calidad que una señal analógica. Veamos la figura siguiente, en donde se representa una



DEGENERACION DE UNA SEÑAL ANALOGICA

señal analógica telefónica típica. En la red telefónica convencio-- nal, la señal es conmutada, atenuada, mezclada con ruido, diafonía y distorsión en la ruta de transmisión, amplificada repetidamente (con todo y ruido), conmutada de nuevo, y así sucesivamente. Mientras más lejos se transmita o sea conmutada, modulada, amplificada, demodulada, etc., mayor es el ruido, la diafonía, los zumbidos y la distorsión. Cuando la señal finalmente llega al receptor y se convierte a sonido, no es ni cercanamente fiel réplica de la voz original.

Una señal digital, en contraste, es virtualmente inmune al ruido, la interferencia y la distorsión, independientemente de la longitud de la ruta de transmisión. Un pulso digital, mientras puede reconocerse como un "uno" o un "cero", puede ser periódica-- mente reemplazado por un pulso nuevo, regenerando así la señal original como lo vemos en la siguiente figura. No obstante debido a la



REGENERACION DE SEÑALES DIGITALES

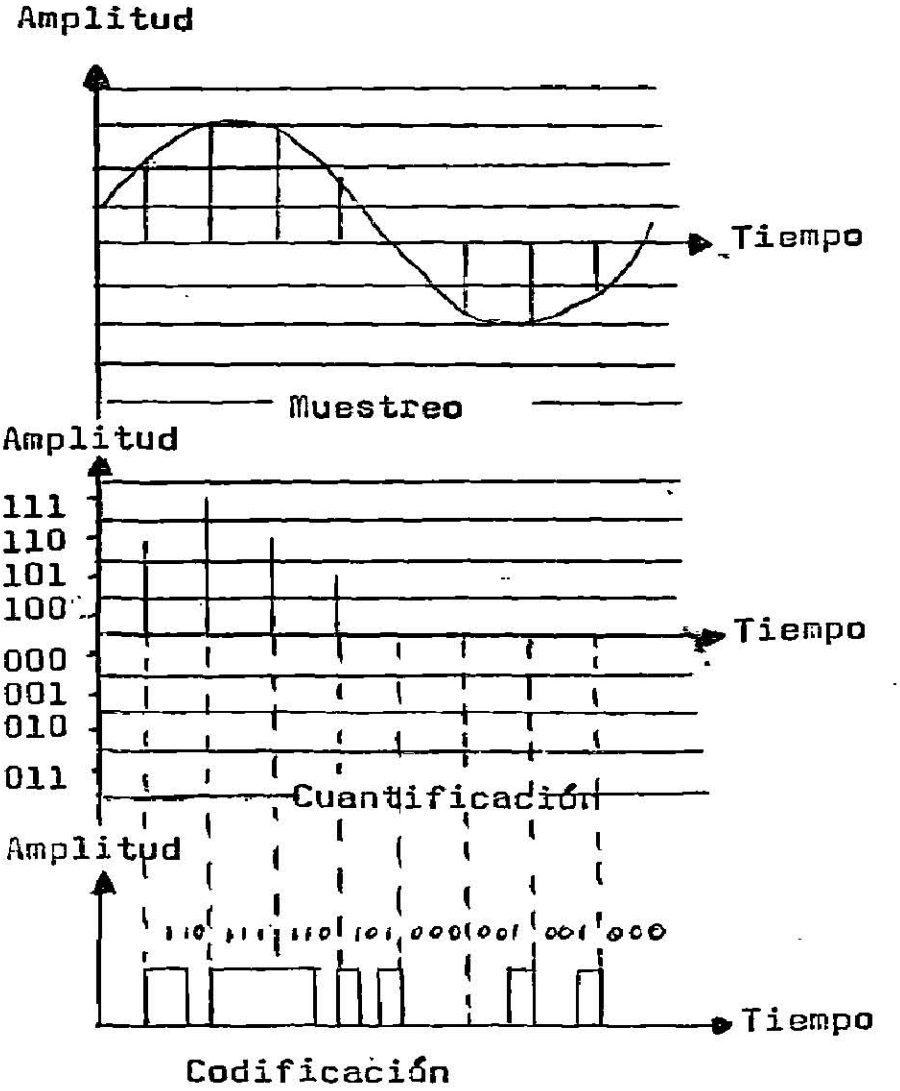
distorsión ocacionalmente se puede perder (o agregar) un pulso, pero la relación de errores es controlable y puede hacerse tan pequeña como se desee.

Otras ventajas son que los circuitos que manejan las señales digitales, por solo necesitar estar conmutando entre dos estados diferentes, son menos complejos que otros y son más prácticos y confiables, y alcanzan mayor velocidad de transmisión.

Todo esto ha llevado a los diseñadores a pensar en medios de pasar a forma digital las señales analógicas, transmitirlas en esa forma y luego reconvertirlas; y a esos equipos se les llama convertidores analógicos-digitales o ADC's y digital-analógicos o DAC's (del inglés Analog to Digital Conversion y viceversa).

B A S I C O

P C M


Amplitud
Tiempo
Muestreo
Amplitud
111
110
101
100
000
001
010
011
Tiempo
Cuantificación
Amplitud
Tiempo
Codificación

P C M

Introducción.

La modulación por pulsos codificados PCM, es un método para convertir información analógica de habla a señales digitales, cada una de las cuales está representada por un tren de pulsos binarios. La conversión se realiza en tres procesos:

Muestreo Cuantificación Codificación

El proceso de elegir los puntos de medición en la curva de conversación analógica se denomina muestreo. Los valores de medición se denominan muestras. Cuando efectuamos el muestreo, tomamos el primer paso hacia una representación digital de la señal de conversación porque los instantes de muestreo elegidos nos dan las coordenadas de tiempo de los puntos de medición.

Las amplitudes de las muestras pueden tomar todos los valores de la gama de amplitudes de la señal de conversación. Cuando medimos las amplitudes de las muestras tenemos que efectuar un redondeo por razones prácticas. En el proceso de redondeo, o proceso de cuantificación, a todas las amplitudes de las muestras entre dos marcas de la escala se les dará el mismo valor cuantificado. La cantidad de muestras cuantificadas es discreta porque tenemos solo una cantidad discreta de marcas en nuestra escala.

Cada muestra cuantificada es luego representada por el número de la marca de la escala, es decir, ahora conocemos las coordenadas en el eje de amplitud de las muestras.

El proceso de muestreo y cuantificación brinda una representación digital de la señal de conversación original pero no en una forma más apropiada para la transmisión sobre una línea o iti-

nerario de radio. Se requiere la traslación a una forma de señal diferente. Este proceso se denomina codificación. Generalmente los valores de las muestras se codifican en la forma binaria, de modo que el valor de cada muestra estará representada con un grupo de elementos binarios. Tipicamente, una muestra cuantificada puede tomar uno de 256 valores. En forma binaria, la muestra estará representada por un grupo de 8 elementos. Este grupo de aquí en adelante se denomina palabra PCM. Para los propósitos de transmisión, los valores binarios 0 y 1 pueden tomarse como correspondientes a la ausencia o presencia de un pulso eléctrico.

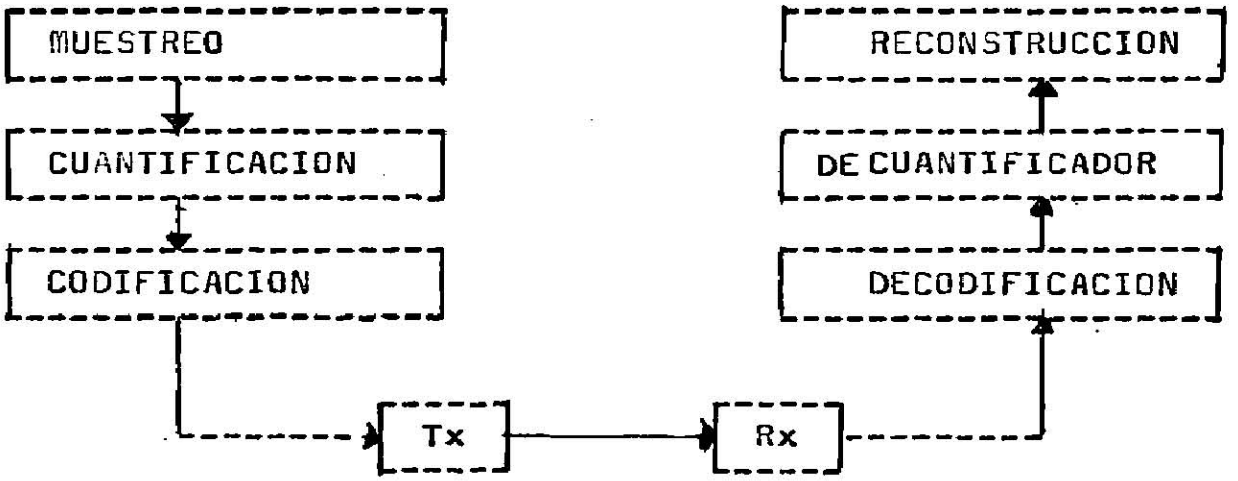
En la línea de transmisión los pulsos de las palabras PCM se distorcionan gradualmente. Sin embargo mientras sea posible distinguir entre la ausencia y la presencia de un pulso, no ha ocurrido ninguna pérdida de información. Si el tren de pulsos es regenerado, es decir, los pulsos frescos a intervalos adecuados, la información puede transmitirse largas distancias con practicamente nada de distorsión. Esta es una de las ventajas de la transmisión digital sobre la transmisión analógica; la información está contenida en la existencia o no de un pulso.

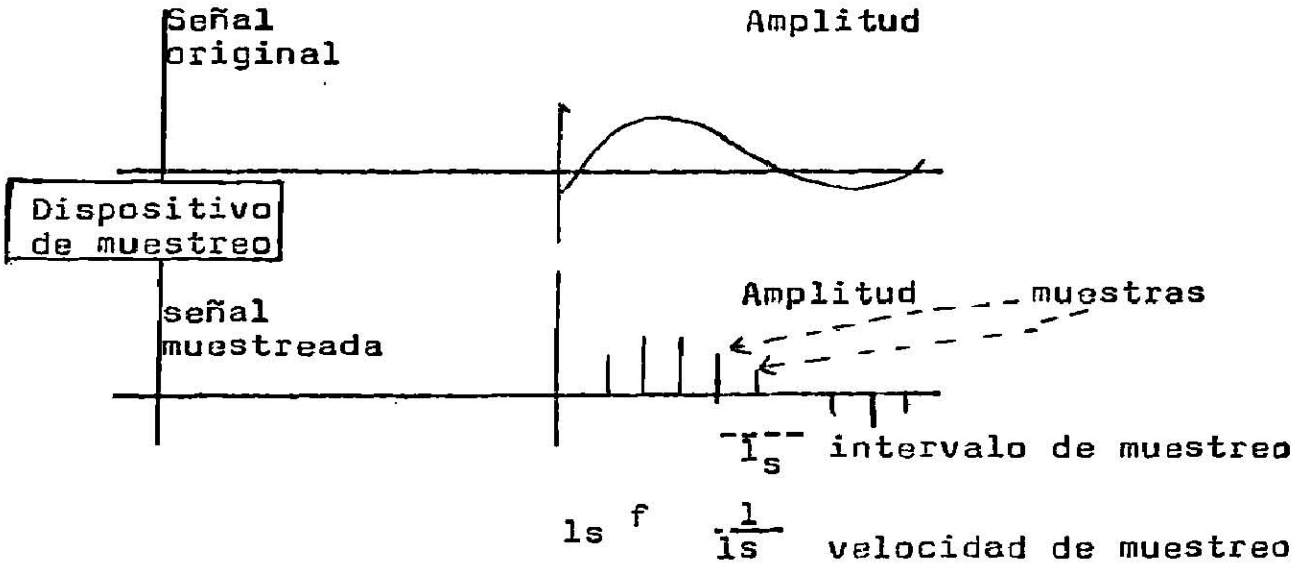
En nuestra imágen del gráfico y la tabla esto es análogo al hecho de que la información de la tabla no es afectada si los dígitos están mal escritos mientras estos sean legibles. Pero si el gráfico está mal trazado, la pérdida de información es inevitable.

En el lado de recepción las palabras PCM se decodifican, es decir son trasladadas nuevamente a muestras cuantificadas. La señal de conversación analógica es luego reconstruída mediante interpolación entre las muestras cuantificadas. Hay una pequeña diferen-

cia entre la señal de conversación analógica del lado de recepción y la señal correspondiente del lado de transmisión a causa del re-- dondeo de las muestras de conversación. Esta diferencia se conoce como distorsión de cuantificación.

Los bloques de funciones en el proceso de modulación por pulsos codificados se muestran en la siguiente figura.





EL PROCESO DE MUESTREO

Estas funciones se tratan con más detalle a continuación.

Muestreo.

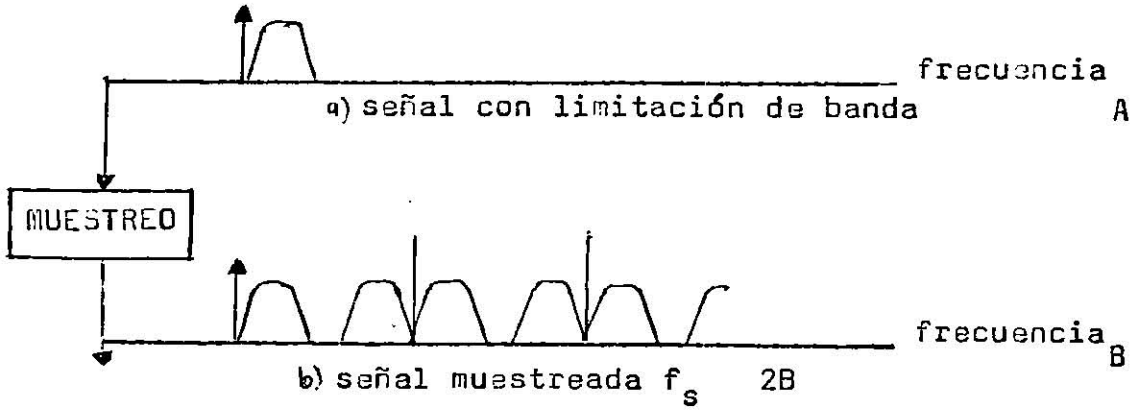
En el significado eléctrico práctico, muestrear es tomar valores instantáneos de la señal analógica a intervalos de tiempos iguales. Veáse figura anterior.

La señal muestreada es un tren de pulsos, cuya envolvente es la señal original.

Ahora, ¿Cuál deberá ser la velocidad de muestreo, es decir, la cantidad de muestras por segundo?. La respuesta a esta pregunta está dada por el teorema del muestreo, que también ilustra el hecho fundamental de que la información contenida en la señal no es afectada por el muestreo:

- La señal original tiene limitación de banda, es decir, no tiene componentes de frecuencia en su espectro más allá de cierta frecuencia B.
- La velocidad de muestreo es igual o superior al doble de B, es decir $f_s \geq 2B$.

El Teorema del muestreo se ilustra en la siguiente figura.



Espectro de : a) Señal con limitación de banda.
b) Señal muestreada.

Obviamente, el espectro de la señal muestreada contiene el espectro de la señal original, es decir no ha ocurrido pérdida de información.

En telefonía, se usa la parte del espectro de conversación entre 300 y 3400 Hz. El espectro de la conversación humana se ex-- tiende desde una frecuencia más baja de alrededor de 100Hz hasta frecuencias de audio muy altas. El aparato telefónico reduce esta gama de frecuencias pero no lo sufucientemente a altas frecuencias, de modo de que a fin de que quede por debajo de este límite de banda a 3400 Hz., la señal de conversación debe pasarse por un filtro pasa bajos antes del muestreo.

En telefonía, se usa una velocidad de muestreo de 8 000 Hz para los sistemas PCM. Esta velocidad es algo superior al doble de la frecuencia más alta de la banda, 3400 Hz.. a causa de la dificultad en la construcción de filtros pasa bajos sufucientemente cortantes.

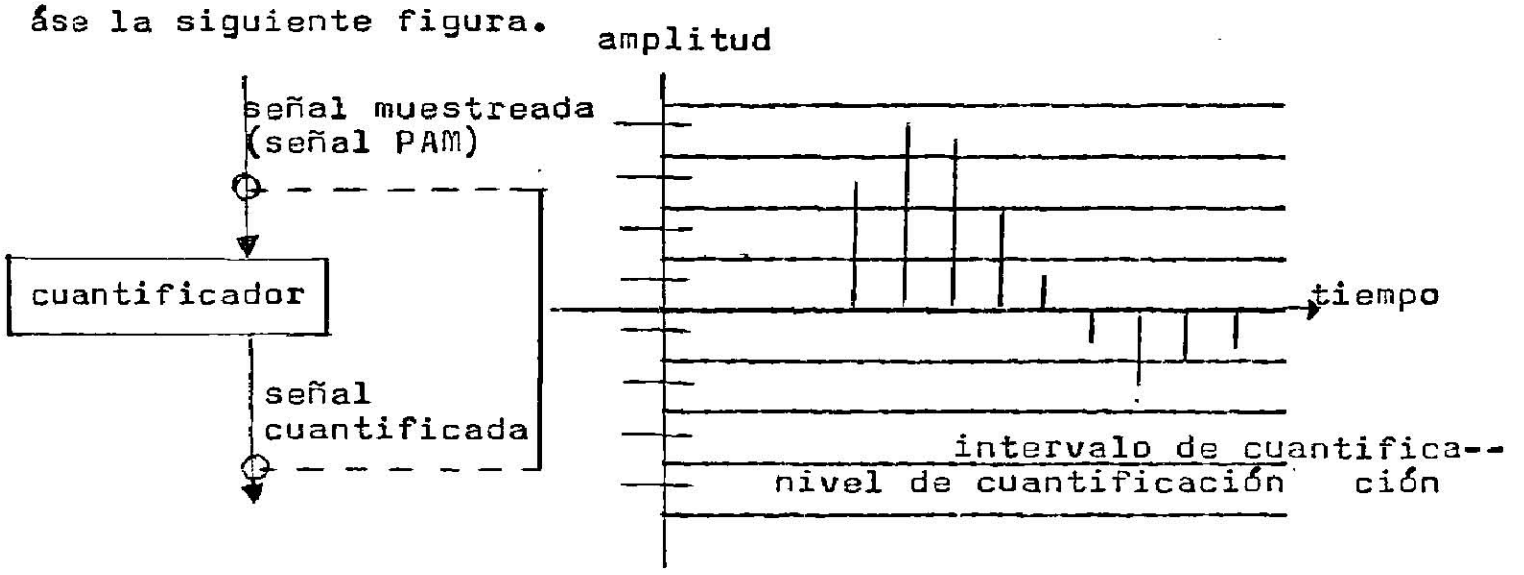
A menudo se dice que la señal muestreada está modulada por la amplitud de pulsos porque consiste en un tren de pulsos, cuyas amplitudes han sido moduladas por la señal original. La modulación por amplitud de pulsos (Pulse Amplitude Modulation=PAM) es un método de modulación de pulsos analógico porque las amplitudes de los pulsos pueden variar de manera continua de acuerdo con las variaciones de la señal original.

La relativa simplicidad de los sistemas PAM los hace atractivos para algunas aplicaciones telefónicas. No obstante, la PAM no es adecuada para la transmisión en distancias largas a causa de la dificultad de la regeneración de los pulsos con suficiente exactitud,

lo cual es importante porque los pulsos PAM contienen la información en la forma del pulso.

Cuantificación.

La gama continua de amplitudes de los pulsos es descompuesta en una cantidad finita de valores de amplitud en el proceso de cuantificación. La gama de amplitudes se divide en intervalos y a todas las muestras cuyas amplitudes caen dentro de un intervalo de cuantificación específico se les dá la misma amplitud de salida. Véase la siguiente figura.



EL PROCESO DE CUANTIFICACION

El redondeo de las muestras provoca un error irreparable, distorsión de cuantificación en la señal.

Este sacrificio voluntario, que puede reducirse a límites bajos adecuados haciendo que la cantidad de niveles de amplitud permitidos sea suficientemente grande, se acepta porque hace posible la transmisión libre de errores teniendo solo una cantidad discreta de amplitudes.

En la figura anterior, la distorsión de cuantificación

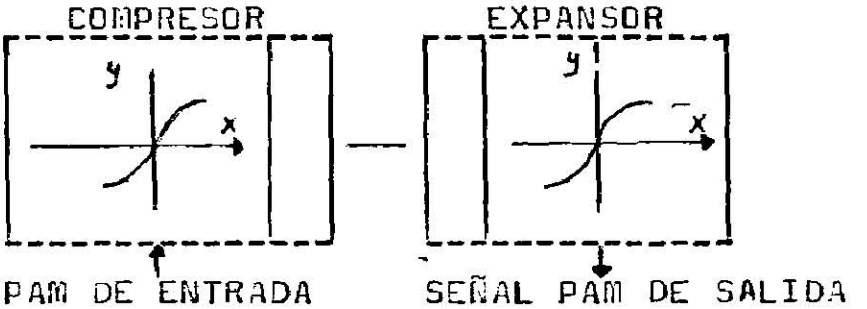
es independiente de la amplitud de la muestra. Esto significa que una persona que habla en voz alta y una en voz baja hacen que el que escucha oiga la misma distorsión de cuantificación. Con respecto a los niveles de conversación, el que habla en voz baja genera mucha más distorsión que el que habla en voz alta. Además, un análisis estadístico muestra que para un hablante individual las amplitudes pequeñas son mucho más probables que las grandes.

A fin de obtener una distorsión de cuantificación aceptable sobre toda la gama dinámica de la señal de conversación, los intervalos de cuantificación deben dimensionarse con respecto a los niveles de conversación bajos, es decir, los intervalos de cuantificación a altos niveles de conversación será mucho menor que la requerida, pero al costo de una gran cantidad de intervalos de cuantificación.

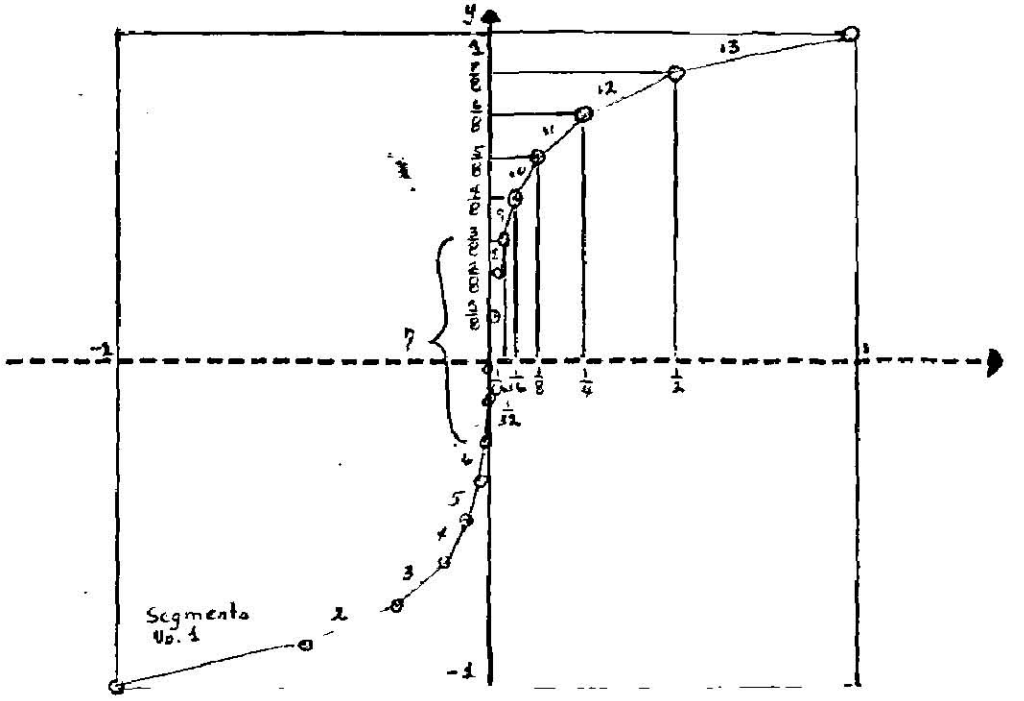
Obviamente, el error de cuantificación no será independiente de la amplitud de las muestras sino que estará relacionado con ella, de modo que las muestras pequeñas estan sometidas a pequeños errores de cuantificación y las muestras grandes están sometidas a grandes errores de cuantificación, a fin de encontrar una solución óptima entre la calidad de la transmisión y la cantidad de intervalos de cuantificación.

Esto puede efectuarse de dos maneras, o comprimiendo el rango dinámico de la señal antes de la cuantificación y expandiéndolo nuevamente en el lado de recepción, o usando intervalos de cuantificación crecientes con la amplitud. Este proceso a menudo se denomina compansión, (Compresión y Expansión). Los sistemas PCM modernos usan el último método de compansión. Con una ley aproximada-

mente logaritmica que gobierna el aumento en el tamaño de intervalo de cuantificación, es posible obtener una relación aproximadamente constante de señal a distorsión de cuantificación en una amplia gama de volúmenes de conversación, empleando a la vez mucho menos niveles que los que se requerirán con intervalos de cuantificación - uniforme. Para la PCM en la telefonía, el CCITT ha recomendado dos leyes, que son conocidas comunmente como la Ley A y la Ley μ. La Ley A se muestra en la siguiente figura.



FUNCION DEL COMPANSOR



La Ley A. La Ley μ es similar pero tiene 15 segmentos.

Estas leyes también se denominan Leyes de Codificación porque en los casos prácticos el proceso de cuantificación se efectúa en el codificador.

Codificación.

Las muestras cuantificadas todavía no son apropiadas para la transmisión, porque sería difícil construir circuitos regeneradores capaces de distinguir entre la gran cantidad de amplitudes de las muestras, usualmente 256, que necesitamos para las señales de conversación.

Sin embargo hay gran flexibilidad en la codificación de estas amplitudes en formas eléctricas adecuadas para la transmisión. En general, la muestra cuantificada puede codificarse en dos o más pulsos con menores niveles de amplitud por pulso. Un grupo de n pulsos, cada uno con B niveles de amplitud discreta posibles, puede representar b^n niveles de muestras cuantificadas. Véase siguiente figura.

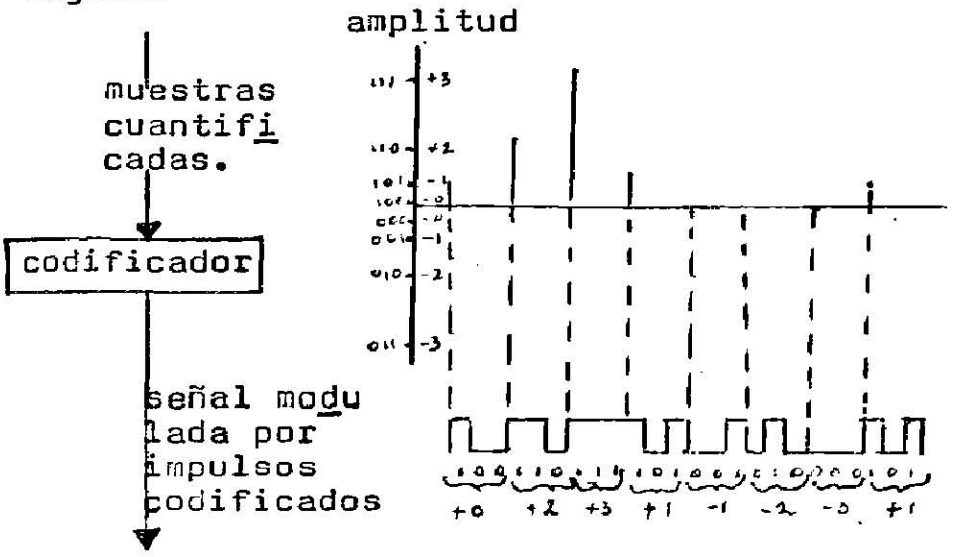
Cantidad de niveles de amplitud b^n	Cantidad de pulsos n	Cantidad de niveles por impulso b
256	1	256
256	2	16
256	4	4
256	8	2

Tabla de alternativas de codificación para muestras cuantificadas con 256 niveles.

Como sabemos, los pulsos con dos niveles, es decir, los pulsos binarios, son atractivos para la transmisión porque son fá-- ciles de regenerar en la línea de transmisión. No es difícil cons--

truir circuitos regeneradores capaces de determinar si un pulso está presente o no.

Los sistemas prácticos actuales usan la codificación binaria de las muestras de conversación cuantificadas. Véase siguiente figura.



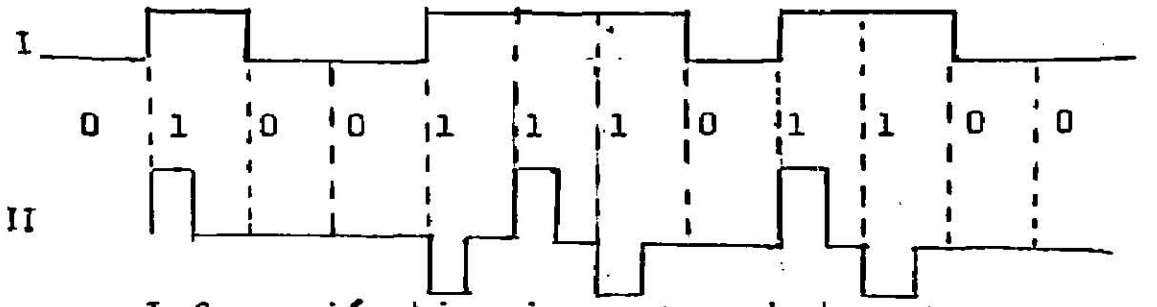
Codificación de muestras cuantificadas con 8 niveles de cuantificación (3 dígitos binarios/palabra de código)

Como la telefonía usa 256 niveles de cuantificación, cada muestra se codificara en un grupo de código, o palabra PCM, consistente en 8 pulsos binarios (8 bits).

Como la velocidad de muestreo usada es de 8 000 muestras/segundo, una señal de conversación modulada por pulsos codificados generará una señal digital de 64 kbit/s.

Transmisión.

Las señales digitales dentro del terminal usualmente se transmiten en la forma de un tren de pulsos unipolares en el modo sin retorno a cero (nonreturn-tozero, NRZ), véase siguiente figura.



Información binaria representada en:

I Un tren de impulsos unipolares sin retorno a cero. (NRZ).

II Un tren de impulsos bipolares con retorno a cero. (RZ)

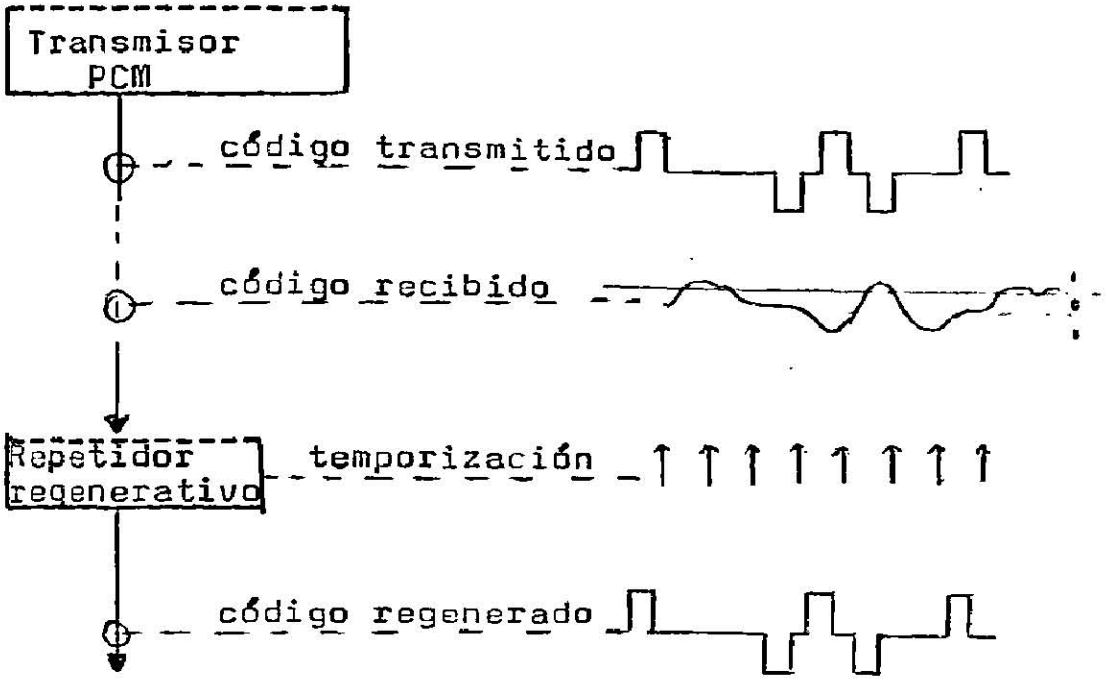
Esta forma de señal no es apropiada para la transmisión en largas distancias. Una mejor forma es una señal bipolar con retorno a cero (RETURN- TO- ZERO, RZ). Las ventajas de esta señal son:

No tiene potencia en las partes inferiores de su espectro, es decir, no tiene componente de corriente continua; esto se debe a las polaridades alternadas de los pulsos.

La interferencia entre símbolos está reducida por la característica de retorno a cero.

Por supuesto, también esta señal será atenuada y distorsionada durante la transmisión y se le agregará ruido a la misma.

En algun punto de la línea de transmisión, la señal debe ser restaurada. Esto se efectúa introduciendo en la línea un dispositivo que primero examina el tren de pulsos distorsionados para ver si el nivel binario posible es 1 ó 0 y luego genera y transmite a la línea nuevos pulsos de acuerdo con el resultado del exámen. Tal dispositivo se denomina repetidor regenerativo, véase siguiente figura.



Formas de los impulsos en una línea de transmisión

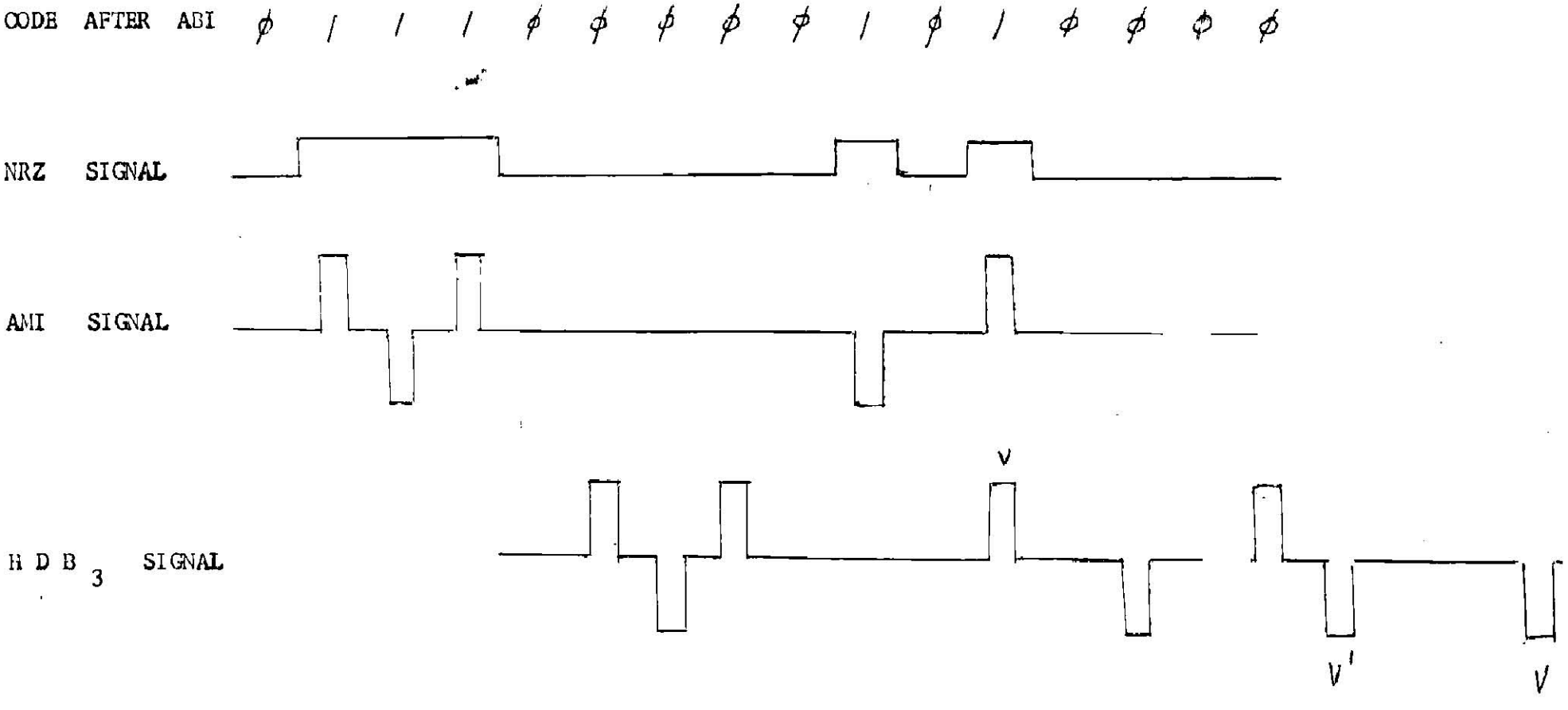
A la vez que se le vuelve a dar forma a los pulsos, se elimina el ruido agregado durante la transmisión, al menos si la amplitud de la señal de ruido no es sufuciente grande como para llevar la señal de código recibida a la zona incorrecta del nivel de decisión de un generador. Normalmente, la señal de código regenerada es idéntica a la señal original. Esta es la razón de la alta calidad de transmisión que se obtiene con los sistemas de transmisión con P C M.

Demodulación.

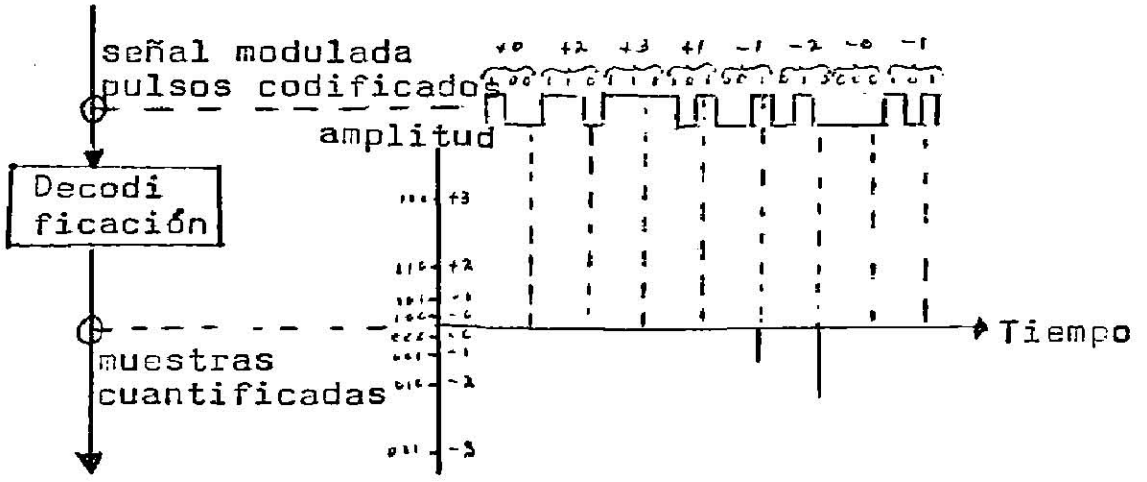
Los procesos de receptor que convierten la señal PCM entrante en una señal de conversación analógica nuevamente son regeneración, decodificada y reconstrucción.

El proceso de regeneración tiene el mismo objetivo y se efectúa de la misma manera que en la línea de transmisión, es decir, los pulsos distorsionados son reemplazados por nuevos pulsos cuadrados, véase figura anterior.

CODIGOS DE TRANSMISION.

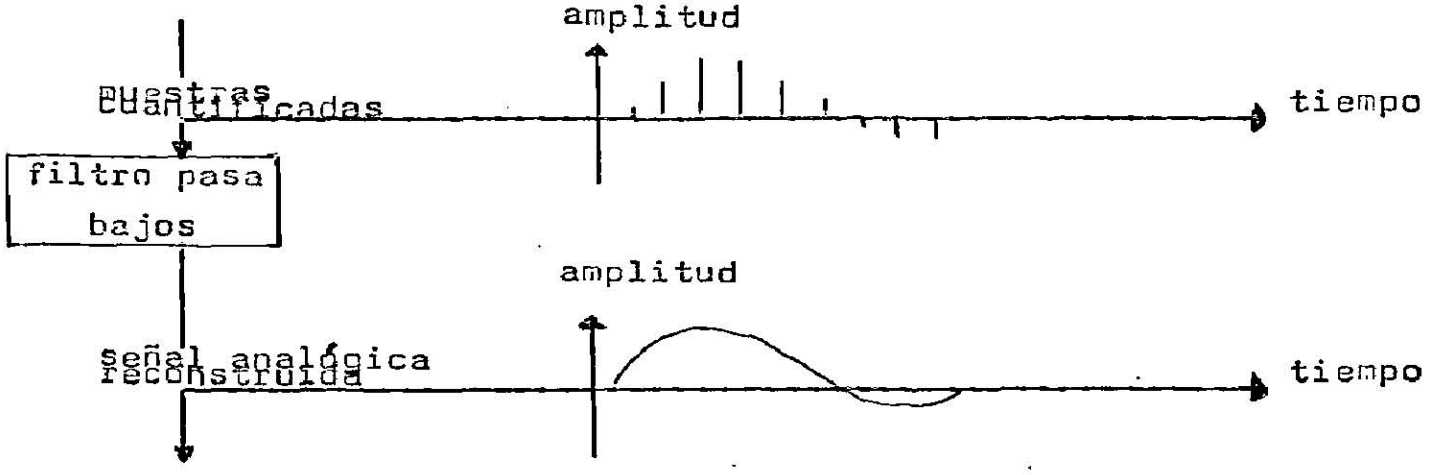


Antes de entrar al decodificador, la señal bipolar es convertida - nuevamente en unipolar. En el proceso de decodificación, las palabras de código generan pulsos de amplitud, cuyas alturas son iguales a las alturas de las muestras cuantificadas que generaron las palabras del código. De modo que después de pasar por el decodificador, se ha recuperado el tren de muestras cuantificadas. Véase siguiente figura.

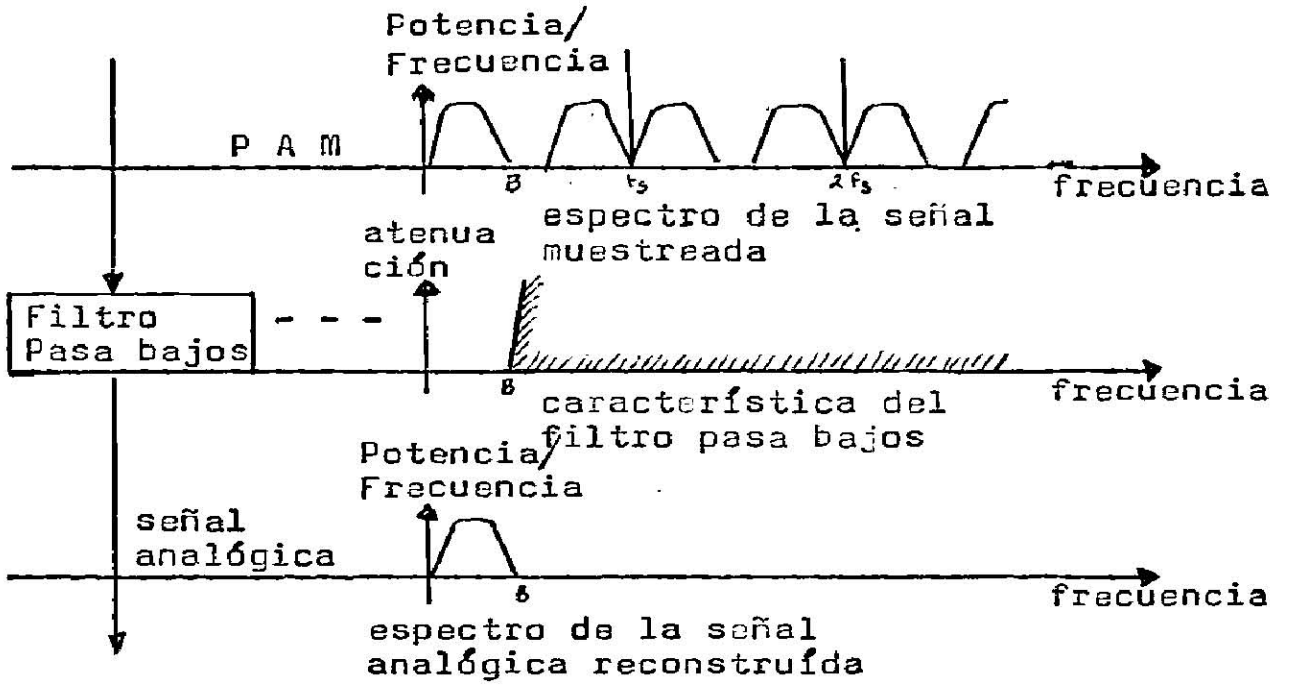


Decodificación de niveles de amplitud codificada.

La señal analógica es reconstruida en un filtro pasa bajos, como se observa en las siguientes figuras.



Reconstrucción de la señal analógica.



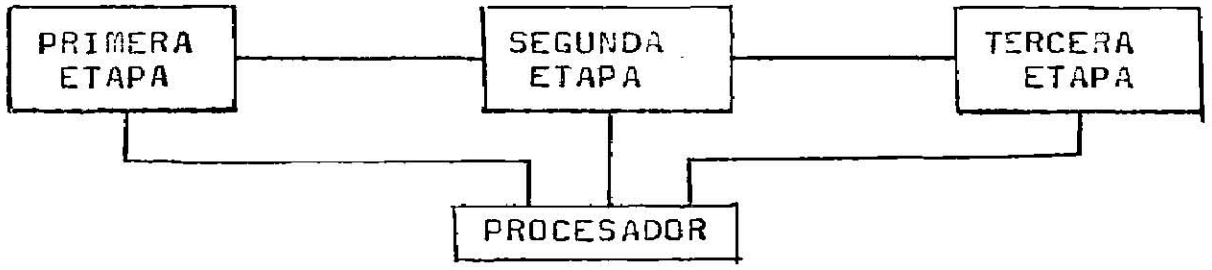
Reconstrucción de la señal analógica mostrada por los diagramas de los espectros.

El espectro de una señal original como se ha mostrado en la figura de la página 30. Un filtro pasa bajos con una frecuencia de corte de B Hz elimina todos los componentes de frecuencia del espectro superiores a B Hz y queda el espectro de la señal analógica deseada.

PROCESO DISTRIBUIDO EN TELEFONIA

Proceso Centralizado.

Fué en los primeros sistemas telefónicos computarizados donde se utilizó el proceso centralizado. El primero que se diseñó, consistía como lo muestra la figura:

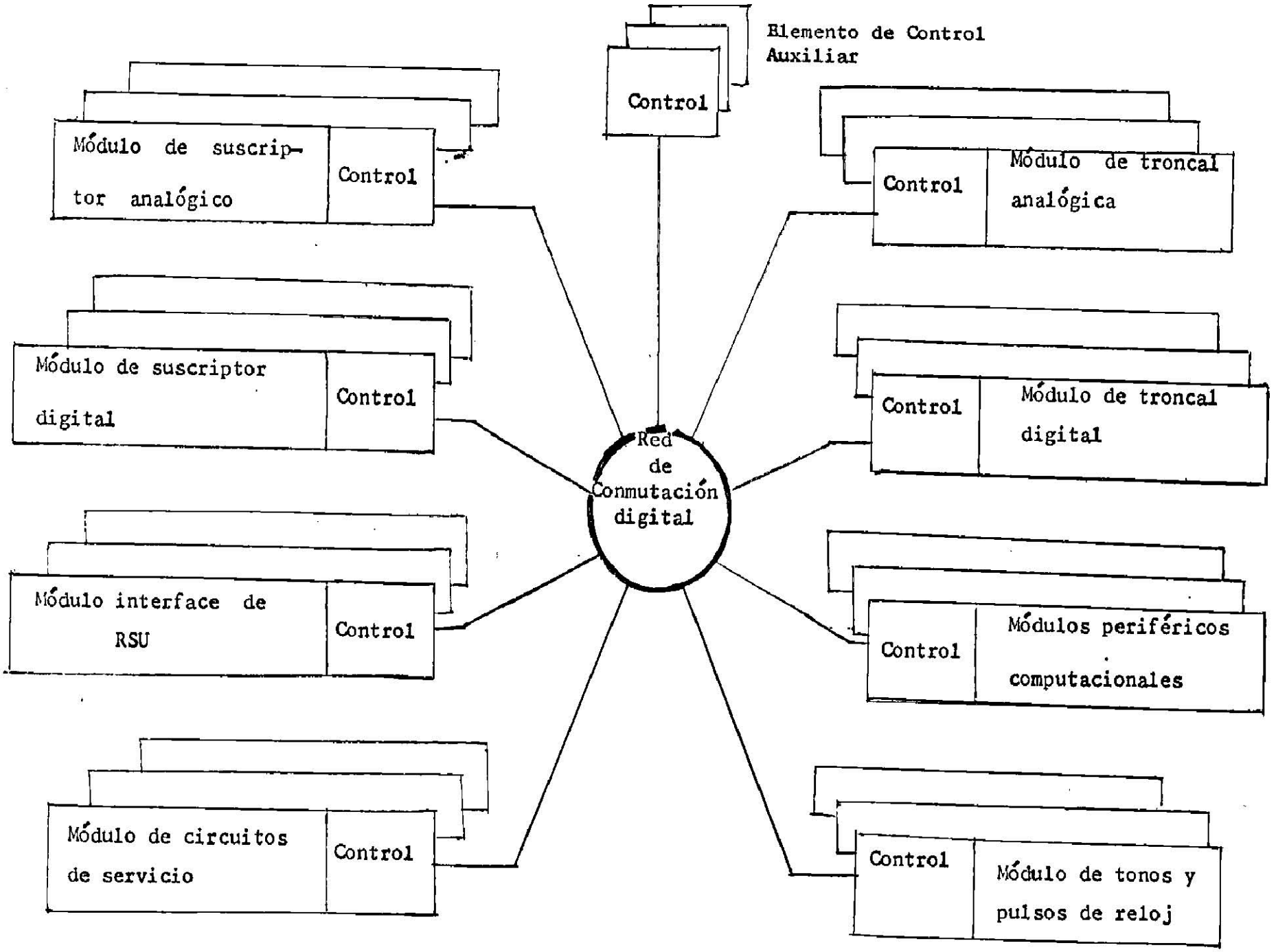


La primera etapa tenía la función de centralizar todas las entradas hacia el procesador y mantener la "comunicación" entre estos. La segunda etapa enlaza la primera con la tercera, aparte de enrutar las llamadas. La tercera etapa tiene la función de centralizar todas las salidas hacia el procesador y mantiene la comunicación entre estos dos.

Proceso Distribuido.

Este proceso nació de la necesidad de un mejor y más efectivo tráfico de llamadas. A continuación veremos en la gráfica un ejemplo de proceso distribuido en telefonía, siguiente página.

Como podemos observar cada módulo tiene su propio procesador (circuito de control), pero cuando requiere comunicarse con otro lo hace a través de la red digital, la cual solo sirve para enlazar a los diferentes módulos cuando éstos lo requieren.


Elemento de Control Auxiliar
Control
Módulo de suscriptor analógico
Control
Control
Módulo de troncal analógica
Módulo de suscriptor digital
Control
Red de Conmutación digital
Control
Módulo de troncal digital
Módulo interface de RSU
Control
Control
Módulos periféricos computacionales
Módulo de circuitos de servicio
Control
Control
Módulo de tonos y pulsos de reloj

MICROPROCESADOR 8 0 8 6

Introducción.

El microprocesador empleado en la central ITT 1240 como elemento de control ya sea terminal o auxiliar es el microprocesador 8086 que es un microprocesador de 16 bits fabricado por INTEL.

En este microprocesador están involucrados todos los procesos terminales del sistema.

El microprocesador puede direccionar directamente 1 M byte de memoria, lo que significa que requiere de 20 bit de direcciones, y a su vez tiene acceso a datos en palabras de 16 bit, tratados como byte de bajo orden y byte de alto orden.

El 8086 es capaz de manejar hasta 64 K puertos de entrada/salida por medio de 16 líneas de direccionamiento, estos puertos pueden ser de 8 ó 16 bits.

Con el fin de permitir la existencia de un bus de direcciones de 20 bits y un bus de datos de 16 bits en un integrado de 40 terminales el bus de datos está multiplexado con los 16 bits menos significativos del bus de direcciones. Las cuatro líneas de direcciones adicionales están multiplexados con información de estados.

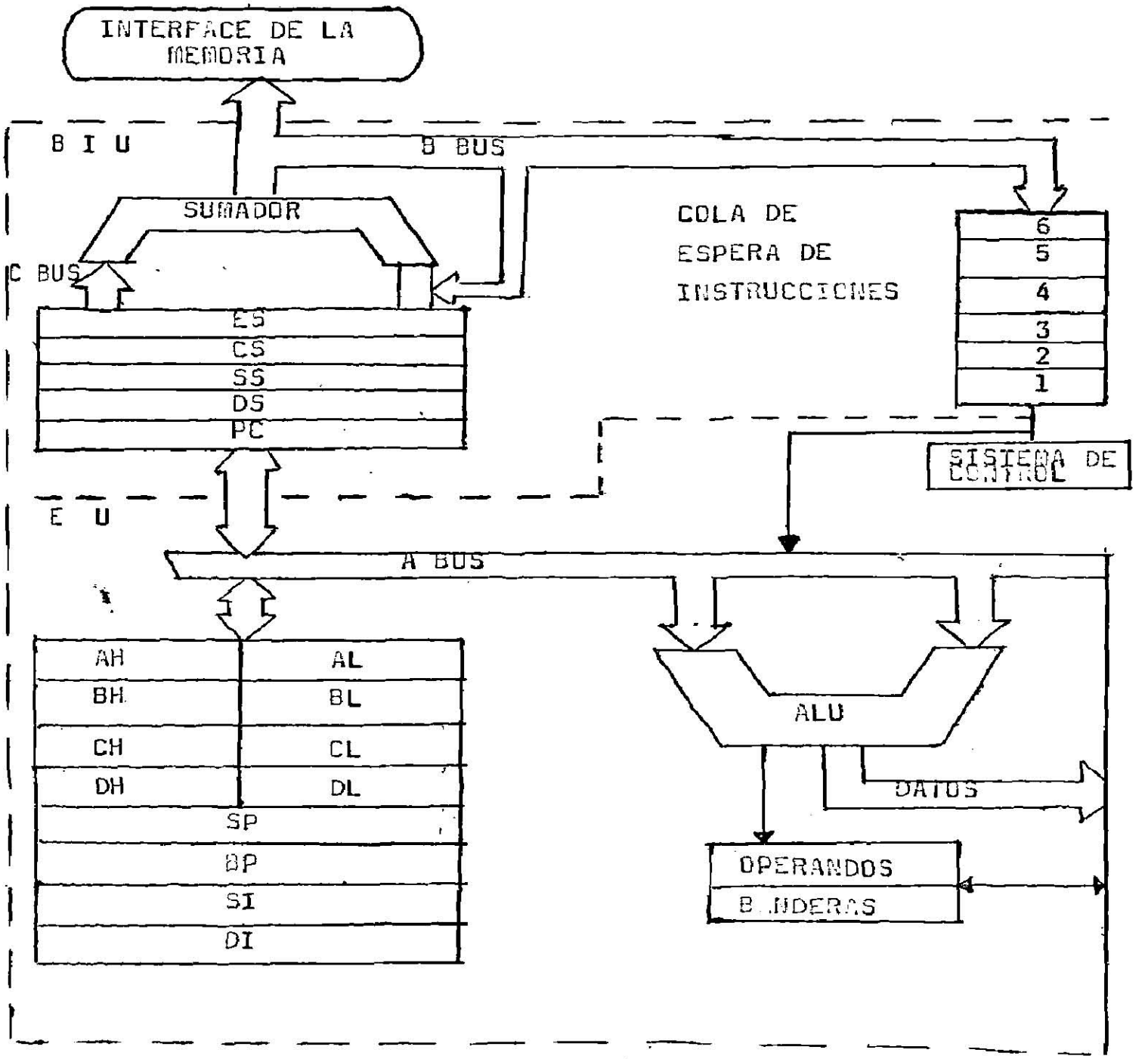
Estas 16 líneas multiplexados dirección/datos, contienen los 16 bits de dirección de bajo orden durante el primer período de reloj del bus, y durante los otros ciclos estas líneas son usadas como bus de datos.

La lógica del CPU del 8086 ha sido dividida en dos partes las cuales son:

La unidad de ejecución (EU) y la unidad de interface del bus (BIU). Estas dos partes operan en forma asincrona.

La EU contiene registros de direcciones y datos, la unidad aritmética y lógica y la unidad de control.

La BIU contiene la lógica de interface de bus, registro de segmentos, lógica de direccionamiento de memoria y la cola de espera de instrucciones de 6 bytes de código objeto. Como se puede ver en la siguiente figura.



DIVISION FUNCIONAL DEL MICROPROCESADOR 8 0 8 6

Arquitectura del 8 0 8 6.

El microprocesador 8086 es un circuito integrado de 40 patas o terminales fabricado en tecnología, Hmos, operando con un reloj de 4 Mhz.

La figura de la página 51 nos muestra este integrado y las funciones de cada una de sus terminales.

Descripción de las terminales:

ADO / AD 15 — Estas son las 16 líneas multiplexados para bus de datos y direcciones.

A16- A17/S3-S4 — Durante el primer período de reloj de la ejecución de una isntrucción estas líneas son líneas de dirección 16 y 17, durante todos los otros períodos proporcionan información que especifica cual registro de segmento está produciendo la porción del segmento de la dirección.

A18 - S5 — Durante el primer período de reloj sirve como línea de direcciones 18, durante los otros períodos de reloj, refleja el estado de las banderas de habilitación de interrupciones.

A19 / S6 — Durante el primer período de reloj sirve como línea de direcciones 19. Durante los otros períodos permanece en estado bajo (0) si se está controlando el sistema de Bus.

$\overline{BHE}$ / S7 — Esta terminal en combinación con Ao son usadas para seleccionar los byte adecuados de la palabra de la memoria, o de entrada/salida.

$\overline{BHE}$	Ao	
o	o	Palabra completa
o	1	Byte superior
1	o	Byte inferior
1	1	Ninguna

$\overline{RD}$ — Es una salida baja cuando el CPU esta leyendo datos de una localidad de memoria o un dispositivo de entrada / salida.

READY — Es usado por un dispositivo de entrada/salida o memoria seleccionada para indicar que esta lista para realizar la transferencia de datos.

$\overline{TEST}$ — Es una entrada usada solamente por la instrucción Wait, cuando esto suceda el microprocesador estará en pausa.

INTR — Entrada para solicitud de interrupciones.

NMI — Entrada para solicitud de interrupciones no mascarables.

RESET — Señal de reset del sistema.

$\overline{SO}$ / $\overline{DEN}$ — Si la terminal MN/$\overline{MX}$ está aterrizada actúa como SO en combinación con S1 y S2 para proporcionar información de estado. Si la terminal esta en " 1 " actúa como DEN

y ésta función está habilitada para transmitir datos.

$\overline{S2}$/(M/$\overline{IO}$) Si MN/MX está en " 1 " seleccionará la función M/$\overline{IO}$ por medio de la cual se podrá accesar a la memoria o bien a una entrada/salida.

Si la terminal MN/MX está en " 0 " nos proporcionará la información de estado según la siguiente tabla:

$\overline{S2}$	$\overline{S1}$	$\overline{SC}$	
0	0	0	Reconocimiento de interrupción.
0	0	1	Lectura entrada/salida
0	1	0	Escritura entrada/salida
0	1	1	HALT
1	0	0	Instrucción FETCH
1	0	1	Lectura de MEMORIA
1	1	0	Escritura de memoria
1	1	1	Inactivo

LSO/ALE Si la terminal MN/MX es " 1 " es usado como ALE (habilitador del latch de direcciones).

LSI/$\overline{INTA}$ Si MN/MX es " 1 " actúa como $\overline{INTA}$ (reconocimiento de interrupción), si MN/MX es 0 funciona como LSI en combinación con QSO las cuales proporcionan el estado de las colas de espera.

QS0	QS1	
0	0	No operación
0	1	El primer byte de una instrucción está siendo ejecutado.
1	0	La cola de espera está siendo vaciada.
1	1	Un byte de la instrucción subsecuente está siendo tomada de la cola de espera.

RQ/GT0 - HOLD

Si MN/MX = 0 la función será RQ/GT0 que es el control de prioridad del bus Local.

Si MN/MX = 1 entonces la función será HOLD que es la línea de solicitud de la función Hold.

RQ/GT1 - HLDA

Si MN/MX = 0 la función será RQ/GT1 para menor prioridad del bus.

Si MN/MX = 1 la función será HLDA que es el reconocimiento de la señal Hold.

$\overline{LOCK}/\overline{WR}$

Si MN/MX = 0 la función será LOCK la cual evita que el procesador pierda el control del bus mientras ejecuta una función.

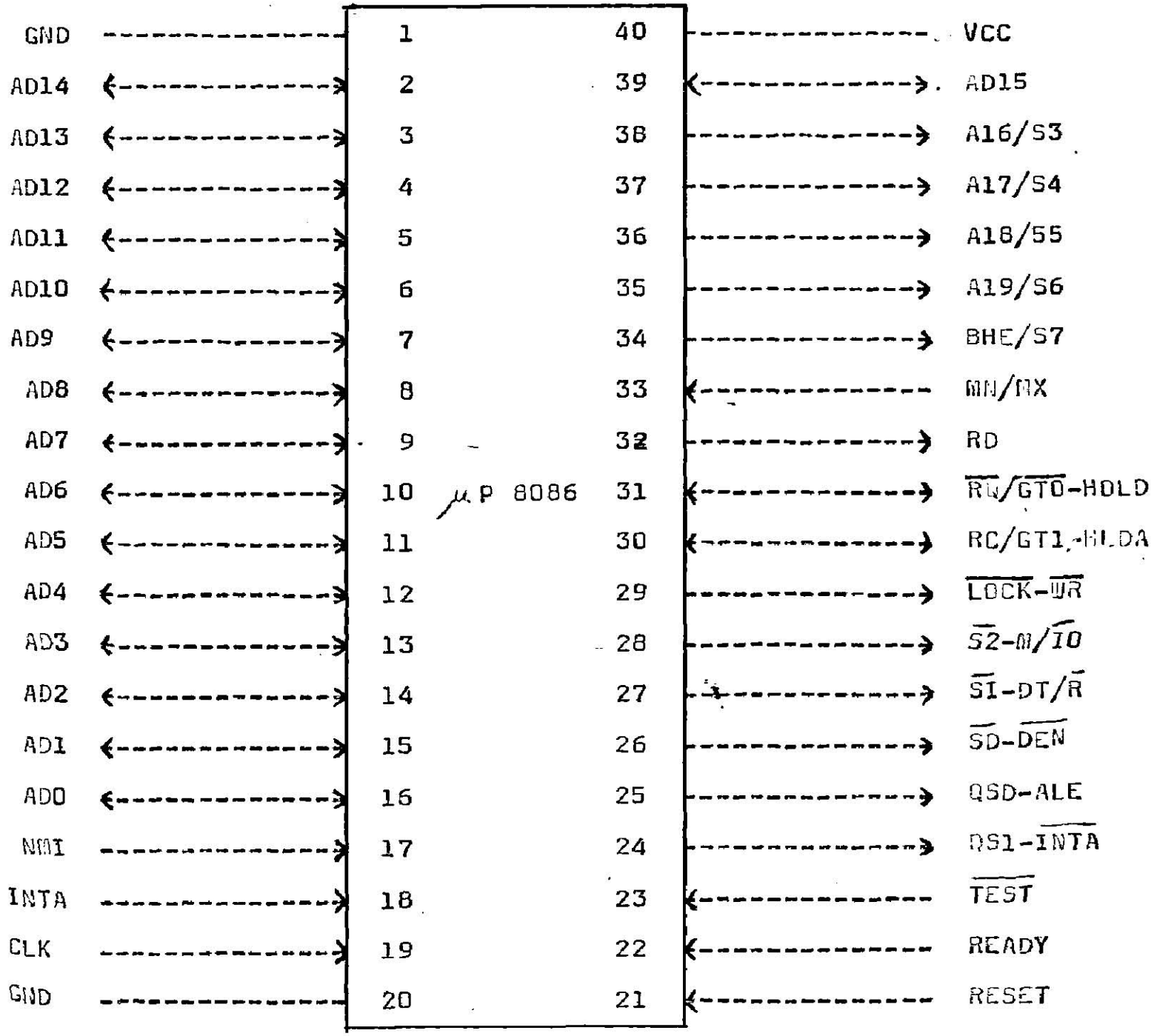
Si MN/MX = 1 seleccionará WR el cual es el control del bus mientras ejecuta una función.

0 máximo modo

MN/MX

1 mínimo modo

CLK Señal de reloj usada para sincronizar toda la lógica del 8086.



DESCRIPCION DE LAS TERMINALES DEL MICROPROCESADOR 8086

DT/R — Cuya función es seleccionar la transmisión o recepción de datos.

VCC — Fuente de poder de - 5V ± 10%

GND — Terminal a tierra.

El microprocesador 8086 contiene 3 grupos de 4 registros de 16 bits y un juego de 9 banderas de 1 bit. Los tres juegos de registros generales, el grupo de registros índices y apuntadores, y el juego de registros de segmentos. Hay además un apuntador de instrucciones que no es accesible directamente, sino que es manipulado con instrucciones de transferencia de control, figura página 52.

Registros Generales.

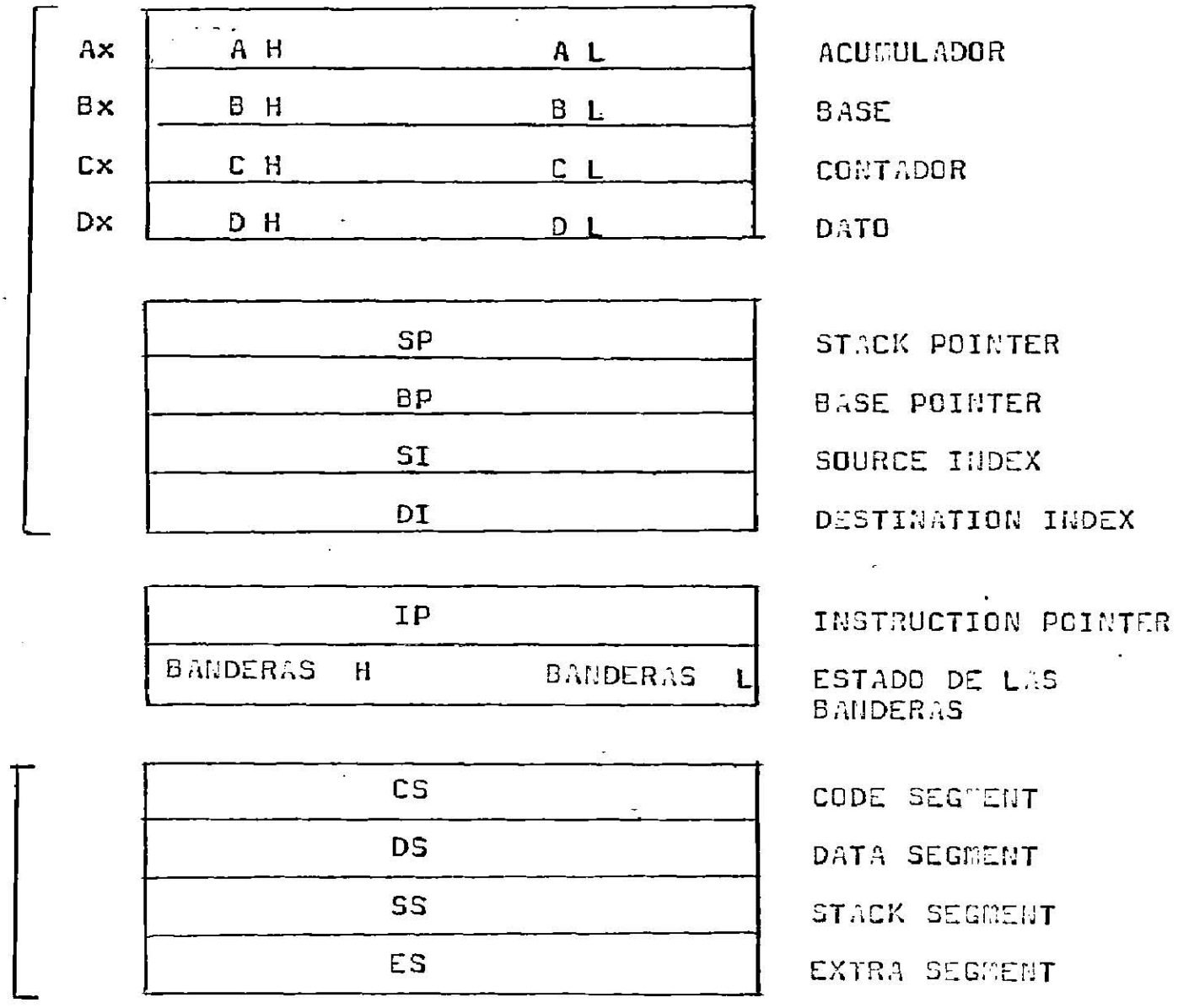
Registros usados para la ejecución de operaciones aritméticas y lógicas en general, los registros son: A X, B X, C X, D X.

Registros Indices y Apuntadores.

Son los registros SP, BP, SI, DI. El grupo de registros apuntadores se utilizan para accesar datos en el "stack" y pueden ser usados como operandos en todas las operaciones lógicas y aritméticas de 16 bits.

El stack pointer (SP) permite la implementación de un stack en la memoria entendiéndose por stack una sección en la memoria en la cual se almacenan datos para los casos en que se utilizan subrutinas dentro del programa principal. Todas las referencias a el SP para direccionamiento de memoria usan el stack segment como registro de segmento.

El apuntador de base (base pointer BP) permite accesar datos en el stack segment.



REGISTROS DEL 8086

Los registros índices son usados para accesar datos en la memoria de datos y son usados extensivamente en operaciones encadenadas.

Registros de Segmentos.

Estos registros están incluídos en todos los cálculos de direccionamiento de memoria. Cada registro de segmento define un bloque de memoria de 64 K byte en la memoria direccionable.

Code Segment (CS).- Cada vez que se va a traer una instrucción de la memoria, el contenido del contador del programa es sumado al contenido del CS con la finalidad de calcular la dirección de la memoria en la que se encuentra la instrucción que se va a buscar.

Data Segment (DS).- Define un bloque de 64 K Byte conocido como segmento de datos, en donde todas las referencias a datos en memoria son tomadas en relación a este registro a excepción de:

a. Para direcciones al stack se usa stack pointer.

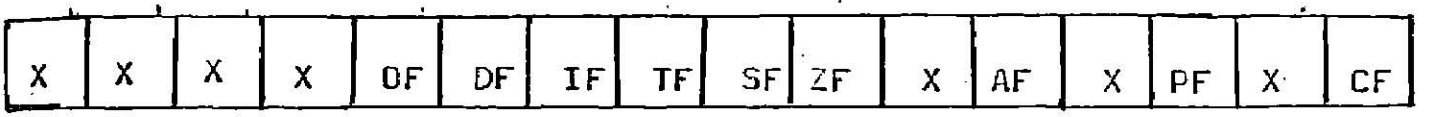
b. Direcciones a datos en memoria usando el apuntador base se toma como referencia el stack segment.

Stack Segment. (SS).- Todas las referencias para el cálculo de direcciones que usan el stack pointer o base pointer se refiere a este registro.

Extra Segment (ES).- El cálculo de las direcciones de memoria para operaciones encadenadas usando el índice de datos tienen como base el extra segment.

Registro de Sanderas.- Las banderas de este grupo son todas de 1 bit; son usadas para grabar información del estado del microprocesador y para controlar la operación del mismo.

Su contenido es el siguiente:



X .- Sin significado

O F .- Sobre flujo, el resultado se excede de la capacidad del registro.

A F .- Carry auxiliar.

P F .- Paridad par.

C F .- Carry, el resultado produce sobre flujo.

S F .- Signo, el bit más significativo del resultado es 1.

Z F .- Cero, el resultado es cero.

D F .- Controla la dirección de manipulación de instrucciones en cadena. (incrementar/decrementar).

I F .- Habilita o deshabilita las interrupciones internas.

T F .- Pone al procesador en modo paso a paso para depuración del programa.

Dispositivos de Soporte y Configuración.

Como sabemos un microprocesador requiere de algunos dispositivos de soporte como son: memorias, generador de reloj, dispositivos entrada/salida, etc.

El generador de reloj debe proveer de ciertas señales adicionales tales como un pulso "reset" que inicialice al microprocesador.

El generador de reloj usado es el 8284 el cual contiene un oscilador de cristal controlado, un contador divisor de tres, sincronizados "ready" para usar un sistema multibus y un "reset" lógico.

Latches (8282-8283) de 8 bits con buffer de salida de 3 estados que son necesarios para los requerimientos de tiempo.

Transmisor Receptor (8286-8287) con salidas de alta impedancia.

Controlador de Interrupciones 8259.- Se usa para controlar la prioridad de las interrupciones hardware enmascarables, este circuito es capaz de manejar hasta 8 interrupciones.

Cuando enviamos un pulso de interrupción, este circuito lo recibe por una de sus ocho entradas, enviándola al microprocesador de acuerdo a la prioridad de la interrupción.

Memoria.

La memoria de un microprocesador está compuesta por una serie de circuitos integrados que no forma parte del microprocesador, sino que es un dispositivo de soporte.

La función de la memoria es la de proveer al microprocesador de un sitio donde almacenar sus programas que va a determinar

la función o funciones específicas del microprocesador en un sistema, es decir, que de acuerdo a las funciones específicas el microprocesador puede realizar las funciones de casi cualquier circuito digital de control.

El microprocesador 8086 tiene una capacidad de direccionamiento directo de un M byte en el cual se localizan, la memoria ROM y los puertos de entrada/salida.

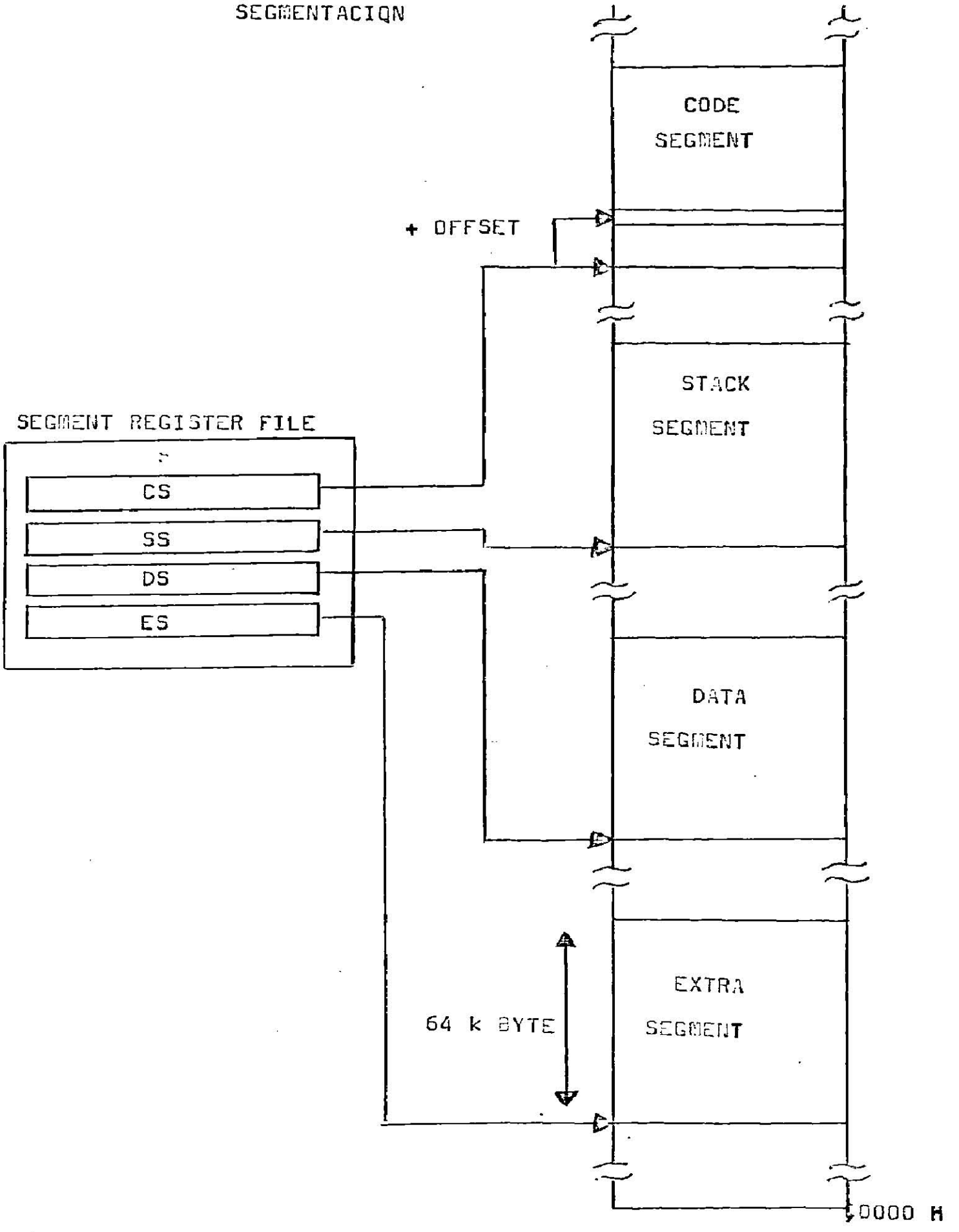
La memoria se encuentra dividida básicamente en cuatro segmentos de 64 K byte cada uno de los cuales son determinados de acuerdo con los registros de segmento, estos segmentos son: segmento de código, stack Datos y Extra, ver figura en la página 57.

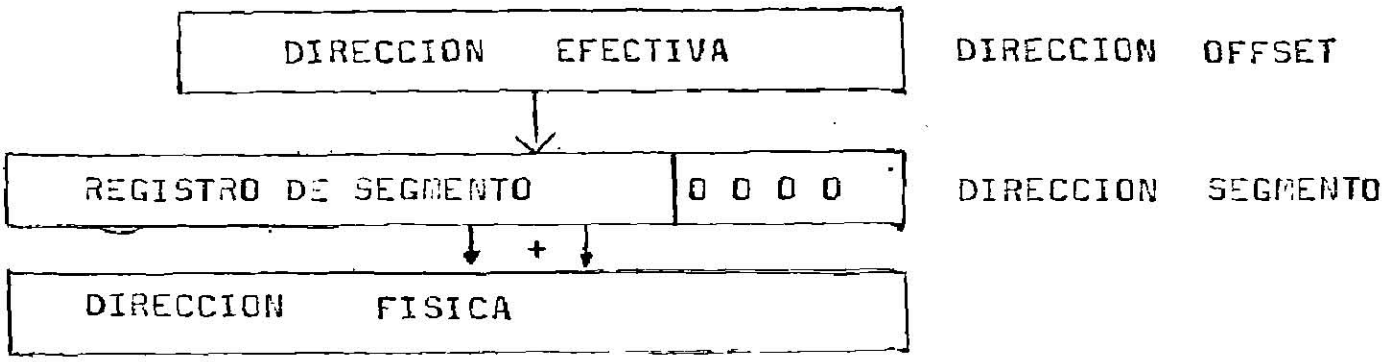
Esta memoria la podemos utilizar como byte o bien como palabra de 16 bits por lo tanto el bus de datos puede hacer transferencias de 8 y 16 bits.

Como ya se mencionó anteriormente el 8086 puede direccionar 1 M bite por lo tanto para direccionar este M bite necesitamos 20 bits en el bus de direcciones, y así obtener esta capacidad de memoria física.

Para poder lograr este direccionamiento se cuenta con los cuatro registros de segmento a los cuales se les agrega cuatro ceros como bits menos significativos y el microprocesador le suma el offset que se ha indicado dentro del segmento, como se muestra en la siguiente figura.

SEGMENTACION





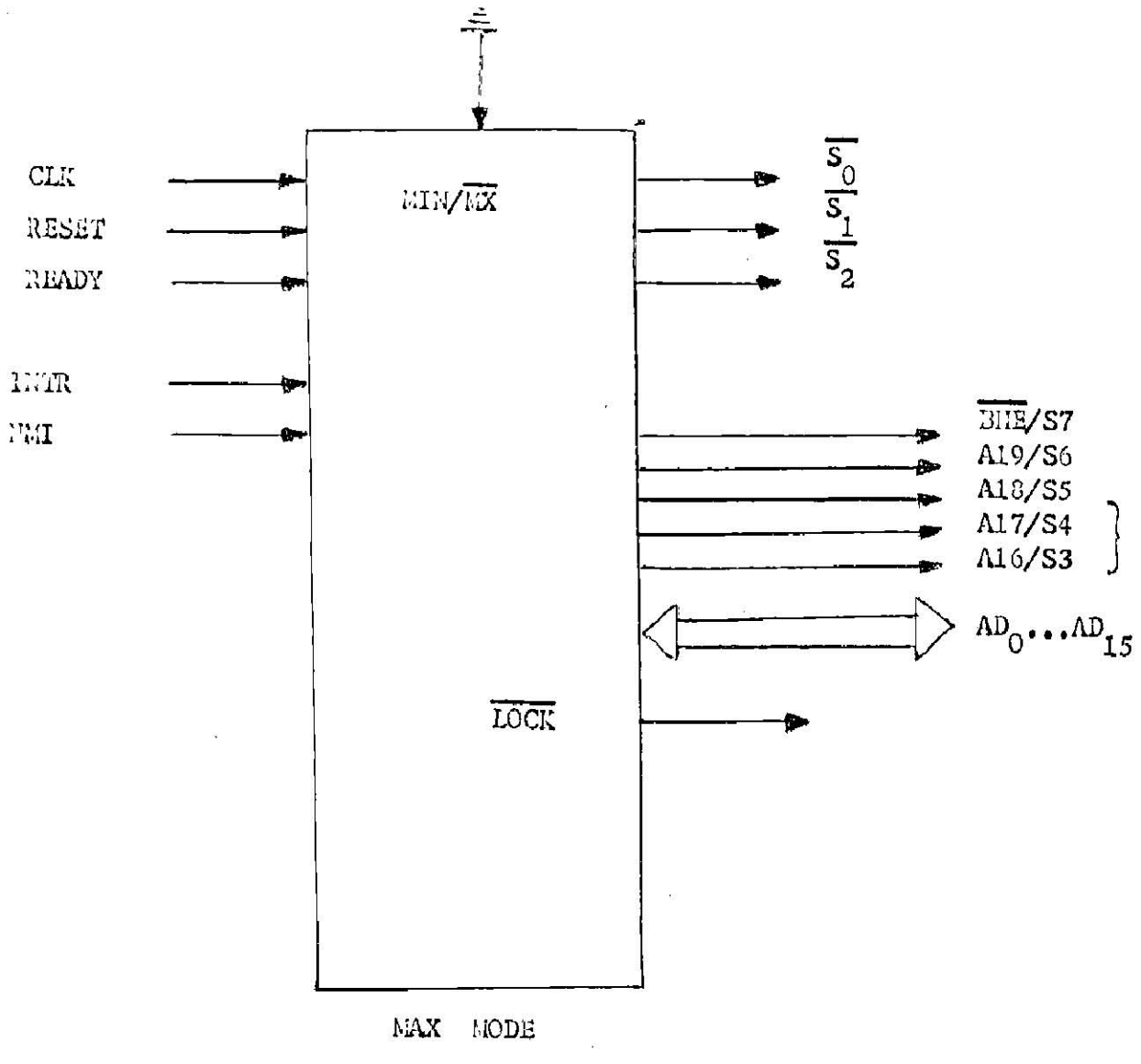
En el microprocesador 8086 contamos con diferentes modos de direccionamiento los cuales son:

- INMEDIATO
- DIRECTO (memoria o registro)
- INDIRECTO (memoria o registro)
- INDEXADO (solo memoria)
- INDIRECTO E INDEXADO (solo memoria)

Este direccionamiento es de acuerdo a la instrucción empleada. La obtención de los datos de memoria puede lograrse ya sea un solo byte o bien la palabra completa dependiendo de las señales Ao y BHE, esto es debido a que la memoria se encuentra dividida en dos bloques de 512 K bytes llamados como byte impar (ODD) y byte par (even) el otro, en la figura de la página 66 se nos muestra gráficamente.

Debido a lo anterior podemos tener alineamiento y no alineamiento de palabras esto es que si tenemos datos de 16 bit en una localidad será una palabra alineada, y en el caso en el cual la palabra se encuentre dividida en dos localidades continuas será una palabra no alineada, para observar lo anterior esta la figura en la página 67.

SEÑALES DEL MICRO EN MODO MAXIMO



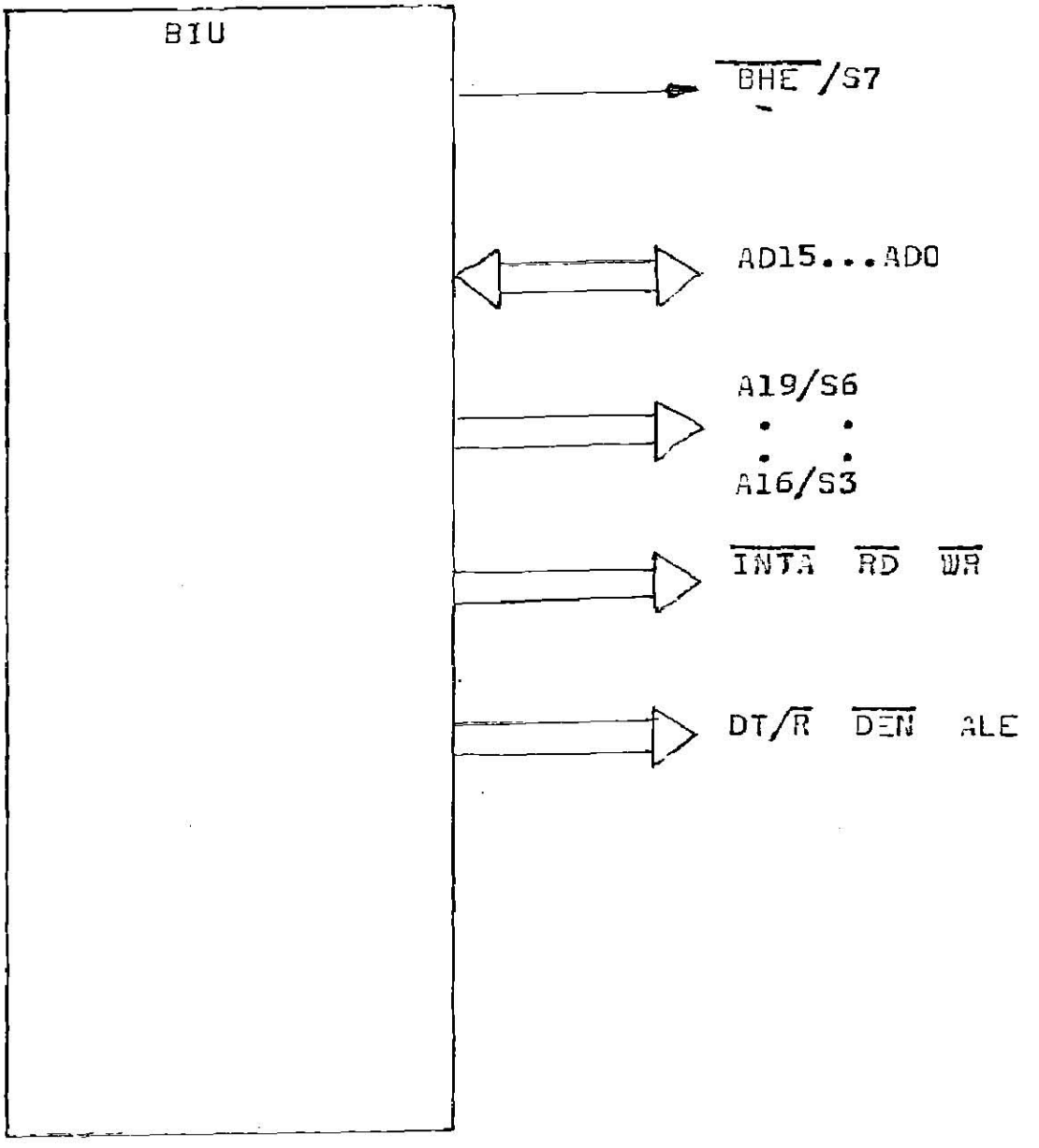
SPARE STATUS BIT = 0

INTERRUPT ENABLE

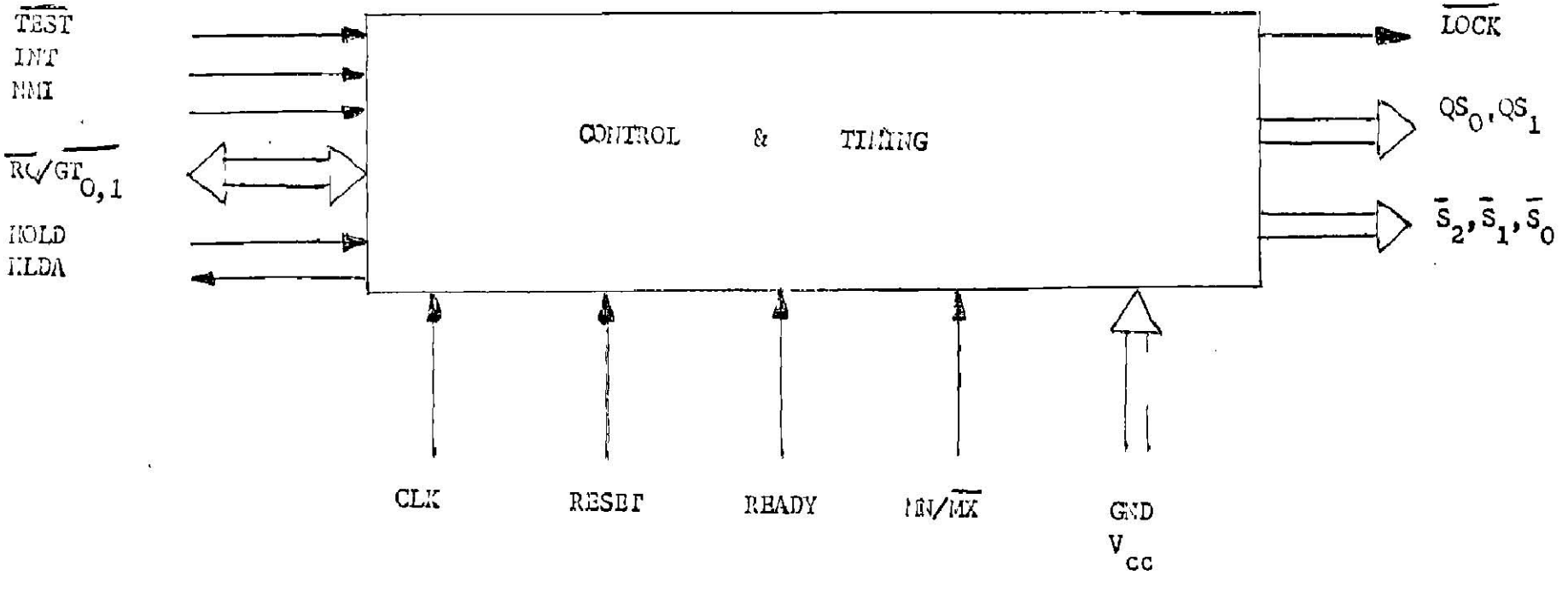
SEGMENT SELECTION

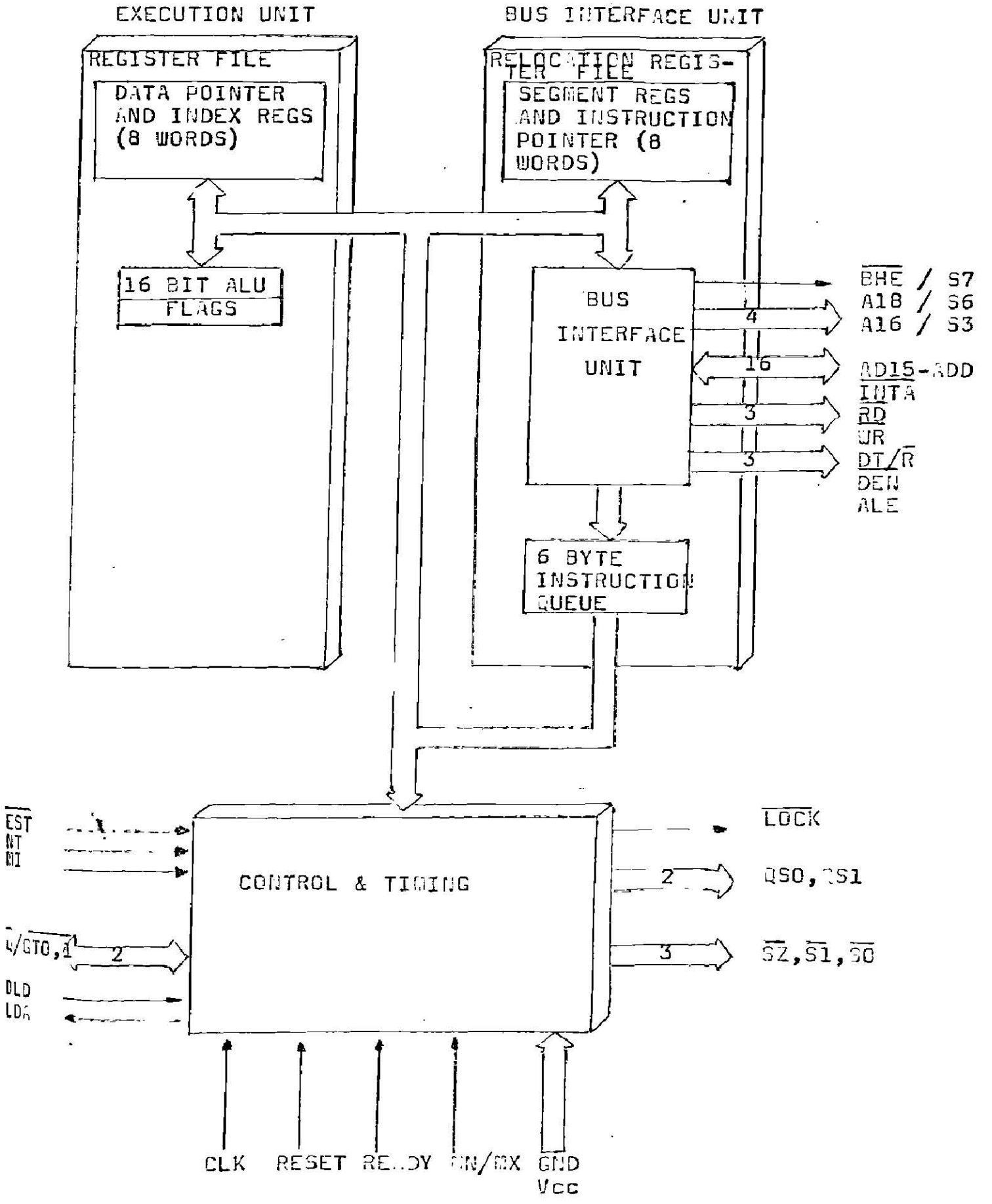
MIN	MAX
ALE	QSØ
$\overline{INTA}$	QS1
$\overline{DEN}$	$\overline{S0}$
DT/$\overline{R}$	$\overline{S1}$
M/$\overline{IO}$	$\overline{S2}$
$\overline{WR}$	$\overline{LOCK}$
HLDA	$\overline{RQ1}$/GT1
HOLD	RQ0/$\overline{GTØ}$

B I U



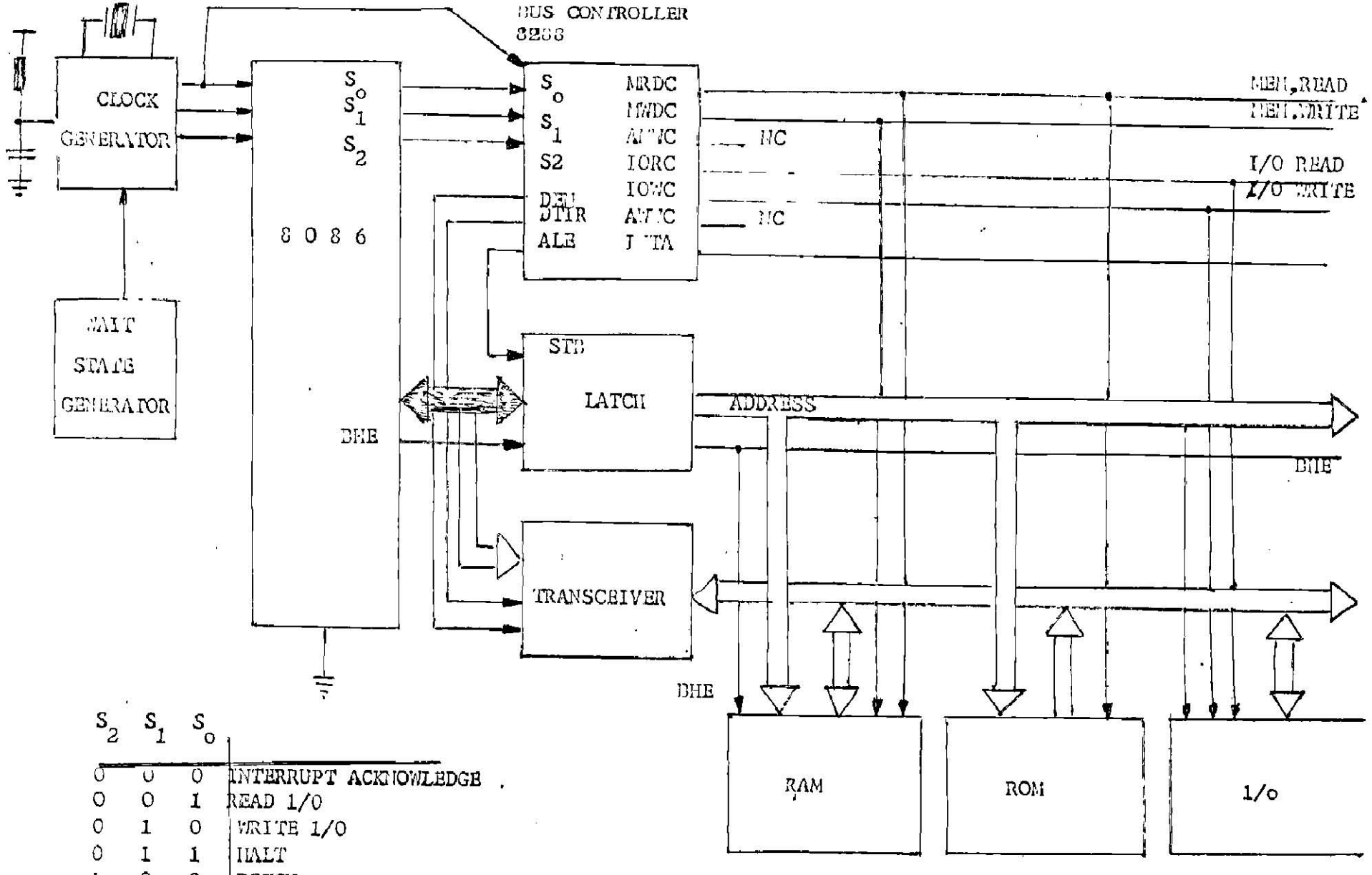
CONTROL Y TEMPORIZACION





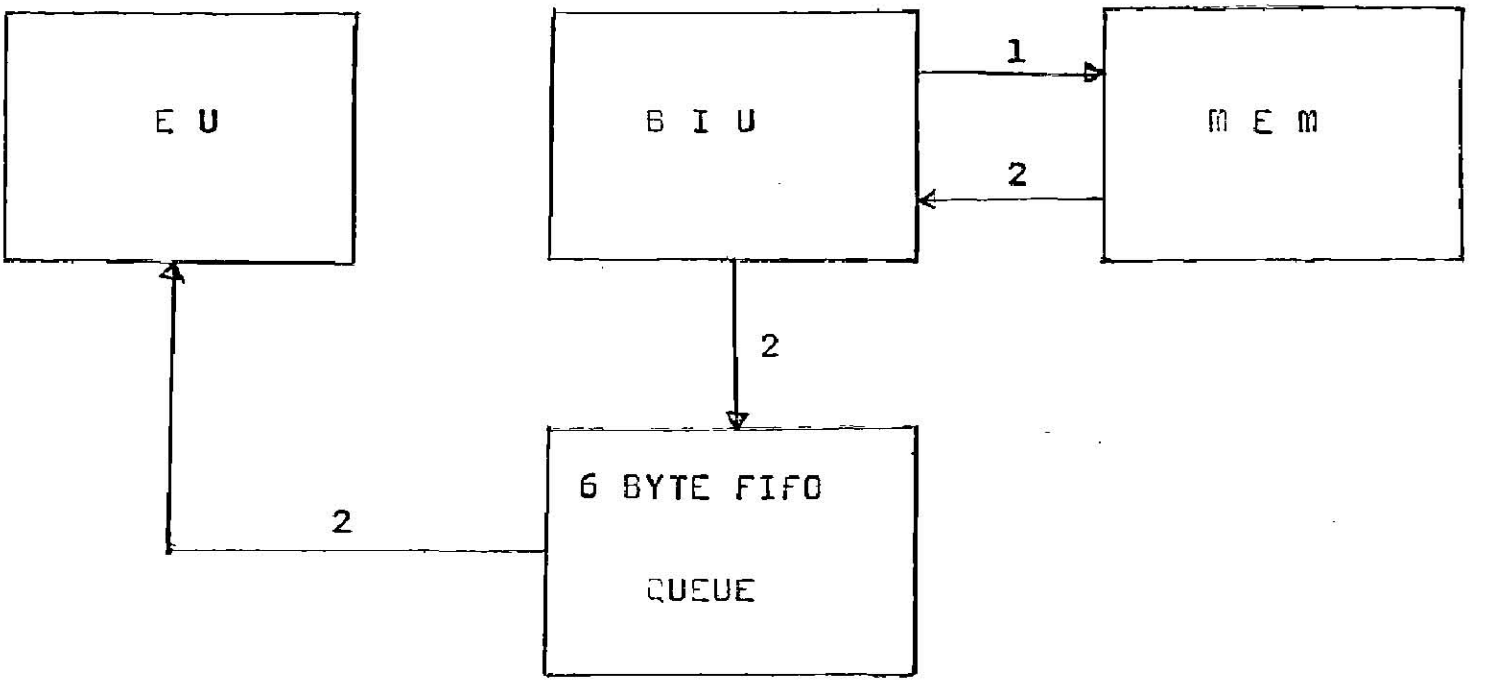
8086 CPU FUNCTIONAL BLOCK DIAGRAM

CONFIGURACION MODO MAXIMO
BUS CONTROLLER
8288



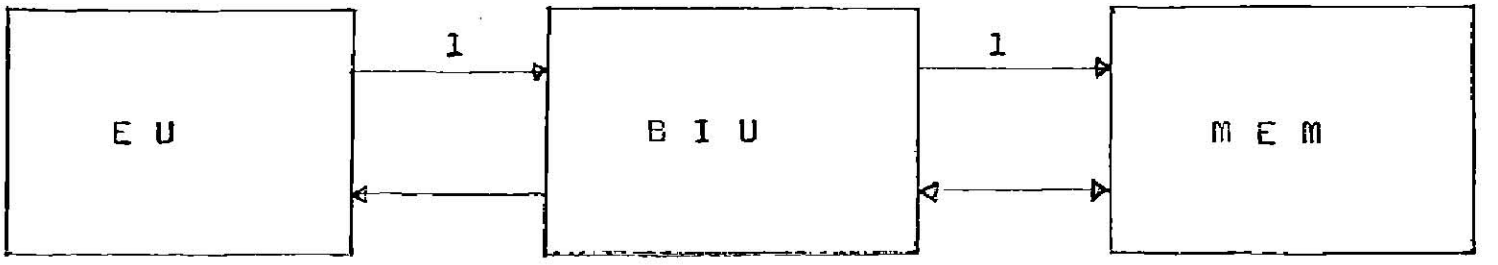
S_2	S_1	S_0	
0	0	0	INTERRUPT ACKNOWLEDGE
0	0	1	READ I/O
0	1	0	WRITE I/O
0	1	1	HALT
1	0	0	FETCH
1	0	1	READ DATA MEMORY
1	1	0	WRITE DATA MEMORY
1	1	1	NO BUS CYCLE (PASSIVE)

OPERACION GENERAL



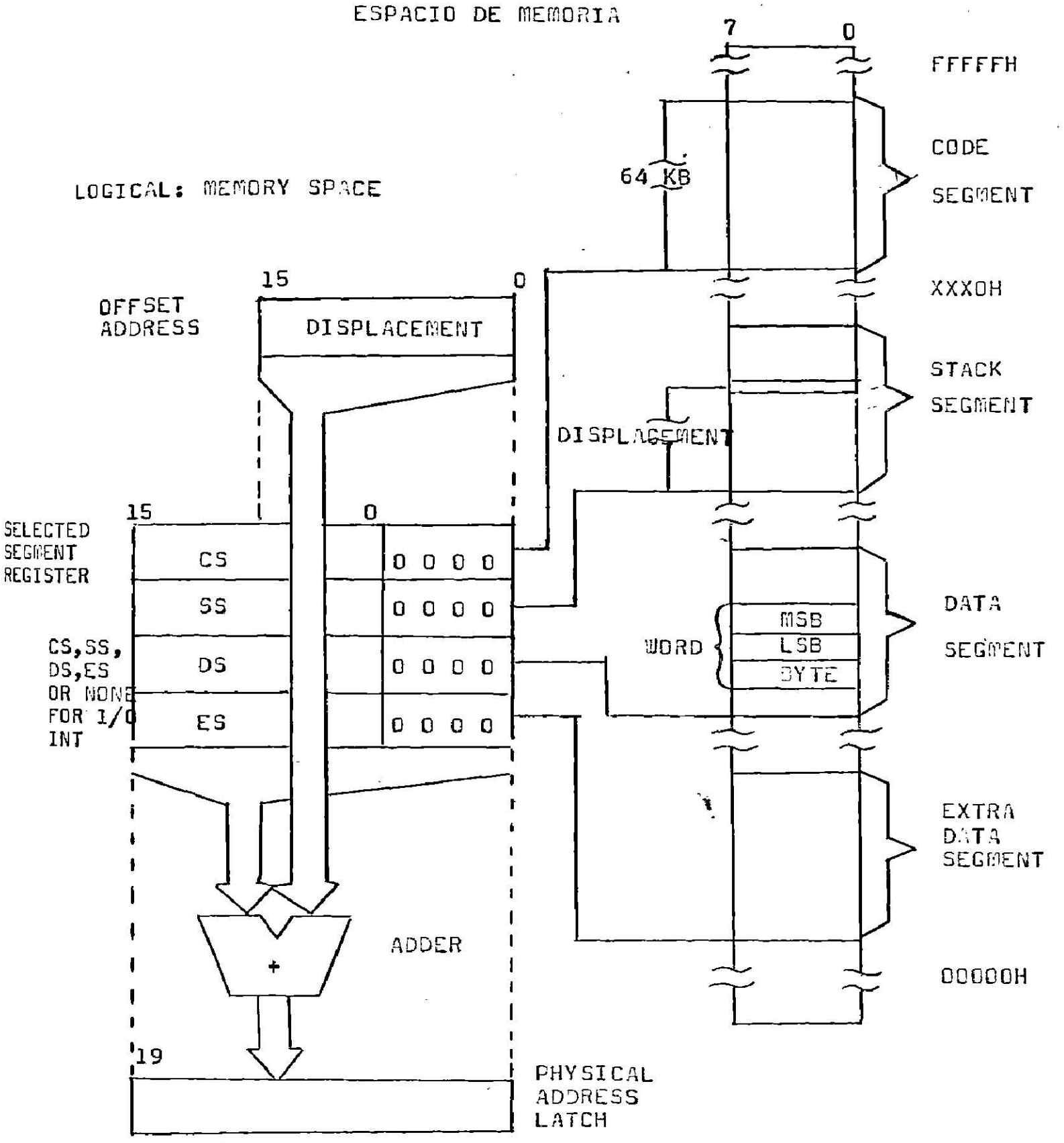
(1) Address for instruction

(2) Instruction flow

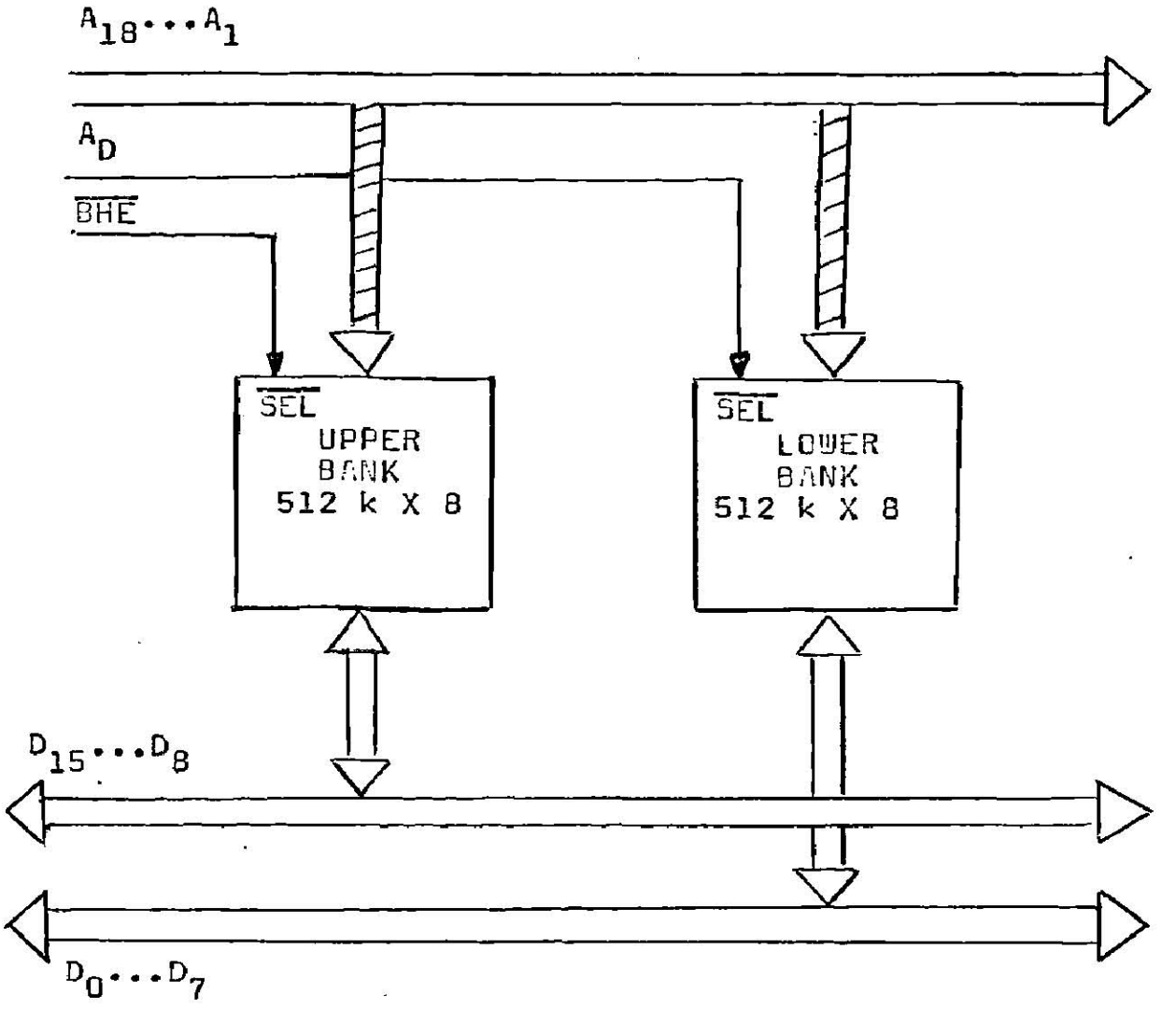


(1) Address for data

(2) Data

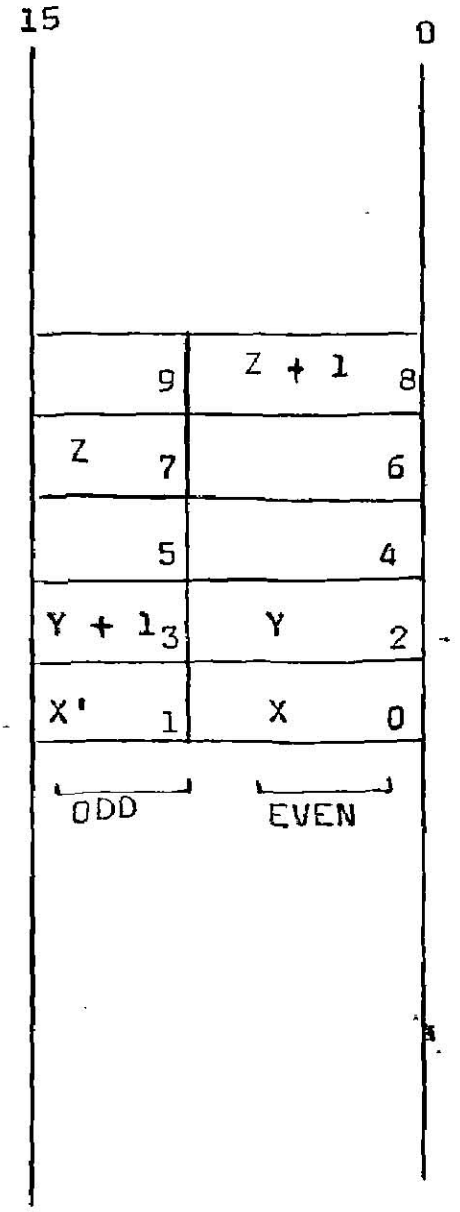

ESPACIO DE MEMORIA
LOGICAL: MEMORY SPACE
7
0
FFFFFH
CODE
SEGMENT
64 KB
XXXOH
STACK
SEGMENT
DISPLACEMENT
DATA
SEGMENT
MSB
LSB
BYTE
WORD
EXTRA
DATA
SEGMENT
OOOOOH
15
0
OFFSET
ADDRESS
DISPLACEMENT
SELECTED
SEGMENT
REGISTER
CS
SS
DS
ES
0 0 0 0
CS,SS,
DS,ES
OR NONE
FOR I/O
INT
ADDER
+
19
PHYSICAL
ADDRESS
LATCH

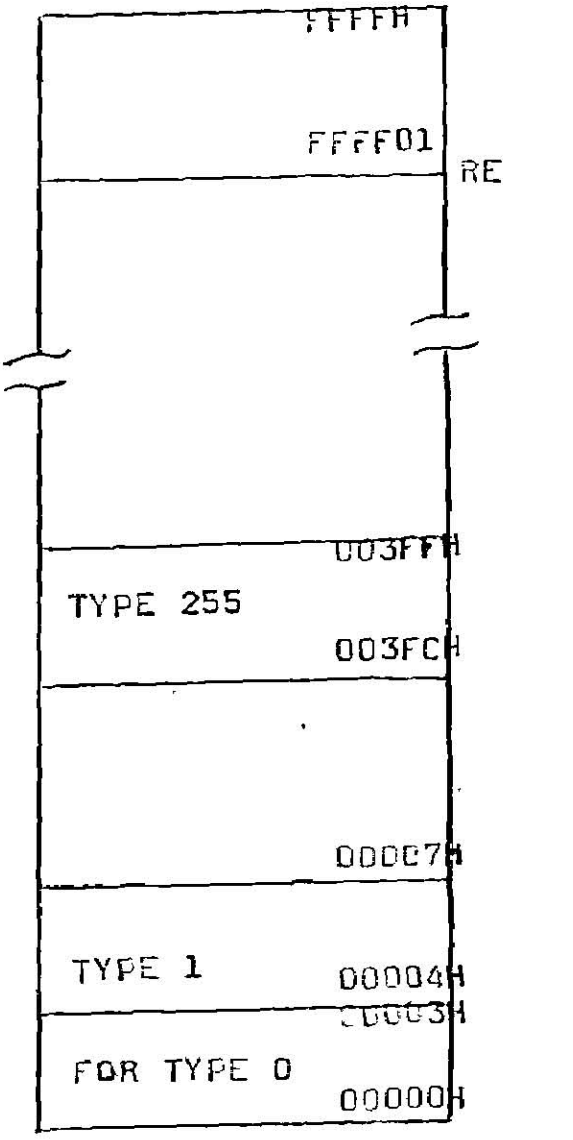
ORGANIZACION DE LA MEMORIA



$\overline{BHE}$	A_0	TRANSFER
0	0	BOTH BYTES
0	1	UPPER BYTE TO/FROM ODD ADDRESS
1	0	LOWER BYTE TO/FROM EVEN ADDRESS
1	1	NONE

ALINEAMIENTO





FIVED MEMORY LOCATIONS

USED FOR 8086

BYTE X ON AN EVEN LOCATION
BYTE X' ON AN ODD LOCATION
WORD (Y,Y + 1) ALIGNED
WORD (Z,Z + 1) NOT ALIGNED

Interrupciones.

A cada interrupción le es asignado un tipo de código para que el CPU lo identifique. El 8086 puede manejar hasta 256 diferentes tipos de interrupciones.

Las interrupciones pueden ser iniciadas por dispositivos externos, en suma, pueden ser disparadas por instrucciones de Software, bajo ciertas condiciones por el propio CPU.

INTERRUPCIONES	PRIORIDAD
Error dividido, INTn INTO.	Alta
NMI	
INTR	
Paso a paso.	Bajo

DEFINICION DE UN FMM (FINITE MESAGE MACHINE)

Un FMM está dividido en dos partes:

. una parte de supervisión.

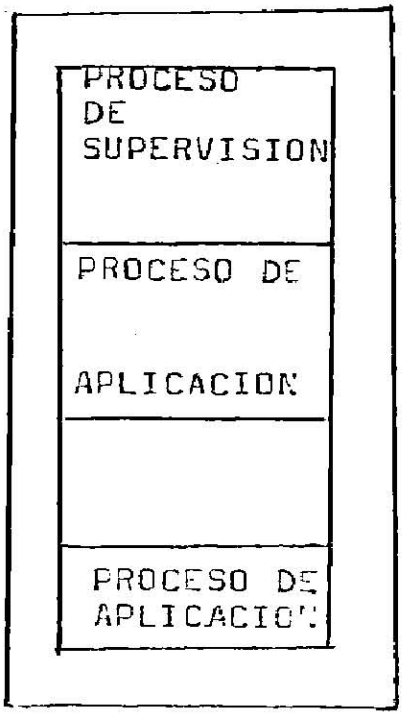
. una parte de aplicación.

Si solamente existe una parte entonces el FMM es considerada una FMM-PROCESO; en este caso la parte supervisora y la de aplicación están incluídas dentro del proceso y solo un proceso puede existir.

Si estos son dos procesos, uno es para la parte supervisora y el otro para la parte de aplicación. La parte de aplicación puede estar basada en la aplicación de multiprocesos entonces es un MULTIPROCESOS-FMM.

El proceso de un FMM opera desincronizadamente con respecto a otros FMM. Este proceso se comunica a través de otros por medio de mensajes.

MONO PROCESO FMM



Esencialmente estos son solamente un tipo de procesos que arrancan su ejecución a la recepción de un mensaje.

Los procesos son ejecutados en base a una prioridad la cual es dada solamente por un indicador de prioridades asociado en cada mensaje.

Parte Supervisora.

Este proceso posee las siguientes características:

- Este consiste solamente de un proceso de definición.
- Este tiene solamente un punto de entrada definida por software. Cuando el elemento de control es esencialmente cargado o cuando el sistema es reestablecido, el SO puede comenzar la ejecución del FMM. En este punto cualquier FMM está inicializando el uso del SO. Esta inicialización debe ser restringida a solamente un FMM relacionado con una manipulación de datos. El wait state puede no estar esperando la recepción del primer mensaje indicando el arranque del trabajo o la recepción del mensaje. FMM-init (si es indicado en el FMM-descriptor) indicando que el FMM está listo para empezar el proceso de supervisión.
- Este puede tener otros puntos de reentrada virtuales de wait state adicionadas dentro del proceso de supervisión.
- Este puede terminar el procedimiento por el uso del primitivo TERMINATE. Cuando el proceso supervisor termina todos los procesos de aplicación pueden ser abortados y si esto es un overlay FMM, pueden ser borrados. Si el FMM es del tipo residente permanentemente, el OS puede

causar la reinicialización de el FMM.

- Este puede enviar y recibir los mensajes básicos y directos.
- Este recibe todos los mensajes básicos enviados al FMM junto con los procesos, entonces directamente transforma este en un mensaje directo y envía este a un proceso de aplicación.
- Cuando necesariamente este inicia la creación del proceso de aplicación para ejecutar las funciones indicadas por la recepción de mensajes.
- Este se comunica a otros procesos enviando mensajes.

Parte de Aplicación.

Este proceso contiene las siguientes características:

- Para multiprocesos-FMM un proceso de supervisión debe ser creado por el proceso de aplicación.
- El primer estatuto ejecutable debe ser el wait case. Si la inicialización de memoria de datos es necesaria o esta debe ser mínima y en un caso no puede el OS primitiva ser invocada.
- Este puede enviar pero no recibir mensajes básicos.
- Este puede enviar y recibir mensajes directos.
- Este puede terminar usando la primitiva OS TERMINATE. Si este termina, todos los mensajes de salida incluyendo cualquier mensaje diferido, pueden ser descartados.
- Este se comunica con otros procesos enviando mensajes.

DEFINICION DE UN SSM (SYSTEM SOPORT MACHINE)

El soporte de muy alto procesamiento activo requiere de un mecanismo operativo eficiente que el soportado por los FMM's. Solamente los sistemas de switcheo son intensamente sensitivos para una señal de tiempo. El tiempo de un SSM debe ser construído en orden, no para afectar adversamente nuestras operaciones. La consideración que el 1240 es una larga integración del sistema, requiriendo algunas diversas funciones de control incluyendo INPUT/OUTPUT (I/O) mecanismos manejadores periféricos, funciones de control de reloj, señal de procesamiento y comunicaciones masivas de soporte, significan que ambos procedimientos, cantidad y diversificación, deben ser contenidos con una estabilidad del mecanismo del SSM.

Cada SSM está considerada como uno o más procedimientos los cuales ejecutan una función de soporte específica para un FMM. Este puede ser un reloj o mecanismo interruptor o manejador de procedimiento o una común subrutina. Que hace esto único, es el hecho que cada uno está altamente activo y en algunos casos depende del tiempo.

El concepto de SSM's implica una reducción en la modalidad y control del sistema y de esa manera debe ser usada con extremo cuidado. Su implementación debe ser restringida a casos donde el tiempo real mande la conducta del sistema a esa existencia.

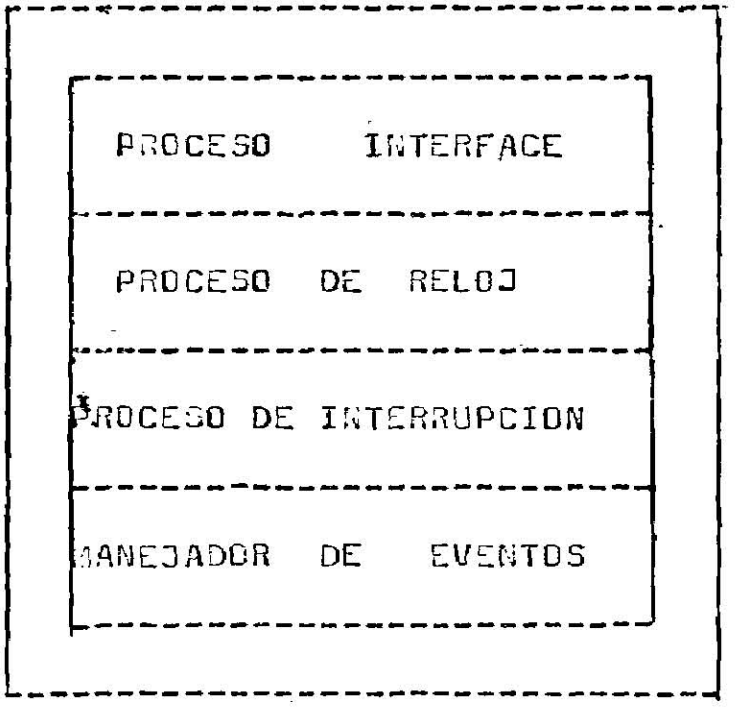
Los SSM's son invocados por los FMM's a través del uso de procedimientos de llamadas al SSM, procedimientos de interrelación o por el OS, a través del uso del reloj o puntos de entrada ininterrumpidos. Si un SSM no puede invocar otros SSM's con la excepción de una SSM, la llamada SSM Data Base especializada. La Data Base

System SSM, es la única dentro del sistema.

Una SSM puede consistir de una o más de cuatro partes componentes: procedimientos de interface, procedimientos de reloj, procedimientos interrumpidos, manejadores de eventos. Todos los componentes deben estar en una sección de datos común. Mientras este consiste de cuatro partes esto no es necesario. Existen SSM's de un solo procedimiento.

Solamente el procedimiento del SSM puede ser ejecutado asincronizadamente, debe ser ejecutado en su diseño y uso.

A continuación se dan a conocer las cuatro fases en que se puede dividir una SSM.



Una SSM está escrita en código puro (Lenguaje ensamblador) para una mayor rapidez en su ejecución.

Proceso Interface.

El proceso interface es el punto de entrada lógico dentro

del SSM. Este puede ser accesado por un proceso de corrida bajo el control de una FMM.

Proceso de Reloj.

Es una función lógica activada periódicamente por una muy alta velocidad (múltiplos de 5 milisegundos hasta 300 milisegundos). Este es usado típicamente para búsquedas en los mecanismos de hardware.

Proceso de Interrupción.

Son muy similares externamente al proceso de reloj. Estos difieren solamente en la manera en la cual ellos obtienen un proceso de control. Estos son activados por software o hardware la velocidad es mayor que un período de reloj básico.

Manejador de Eventos.

Los procesos de interrupción y reloj pueden en algunos casos tener excesivo trabajo. Si el trabajo es ejecutado mientras la interrupción del sistema está deshabilitado el período del búsqueda y el reloj y con el sobreflujo puede ocurrir la degradación del sistema de esta manera el trabajo que está hecho no necesita ser hecho con interruptor deshabilitado, debe ser ejecutado en un manejador de eventos.

Esta restricción especificamente se aplica para la transmisión de mensajes, como ello requiere una considerable cantidad de recursos del O S y tiempo.

SISTEMA OPERATIVO

Funciones. Diferentes Componentes.

El sistema operativo es la parte del software distribuído en los diferentes CE's del sistema, que permite que las FMM's y las SSM's puedan existir. Es decir, permite la comunicación entre FMM's mediante mensajes, independizándolas de las funciones que éstas tienen en el sistema, creando y acabando los procesos y permitiendo que estos se ejecuten. Trata las interrupciones de reloj y periféricos para hacer entrar las rutinas de las SSM's adecuadas, así como los manejadores de eventos que aquellas puedan necesitar, etc.

Por otra parte, el S O se encarga del gobierno de la red de conmutación, la interfaz terminal, y todos los periféricos clásicos no telefónicos (disco, cinta, etc.)

Finalmente se encarga de la recarga y recuperación, debido al fallo, de los diferentes elementos de control.

Las funciones que realiza el sistema operativo son las siguientes:

Gestionar el tiempo del procesador.- Es decir, indicar en cada momento qué programa tiene que ejecutarse en cada elemento de control. Para ello existen una serie de prioridades asignadas a las diferentes cosas a hacer, que se discutirán posteriormente.

Gestionar la memoria de los diferentes elementos de control y de masas.- Es decir, proporcionar a los programas que lo necesiten, la cantidad de memoria necesaria; por ejemplo al crear un proceso, el sistema operativo le proporciona una zona donde el proceso anotará sus datos locales. Al ser dinámicos los procesos (se crean y se destruyen) cuando estos mueren, el sistema operativo re-

cupera esta zona para podérsela signar a otros nuevos procesos.

Controla también la memoria necesaria para el intercambio de información entre procesos, proporcionando los buffers necesarios para almacenar la información.

Así mismo el sistema operativo controla las zonas de memoria de reserva donde se cargarán los programas de overlay (los no residentes) desde el disco, etc.

Controlar el terminal interface (memoria, puertos y canales del mismo) y la red de conversación. Gracias a ello puede establecer caminos en la red con la estrategia de establecimiento vista en la descripción del hardware. Controlando la memoria, los puertos y los canales del TI permite el envío y recepción de mensajes. Finalmente realiza el "cut-through" para conectar los diferentes terminales a la red, permitiendo así el paso de la conversación a través de la misma.

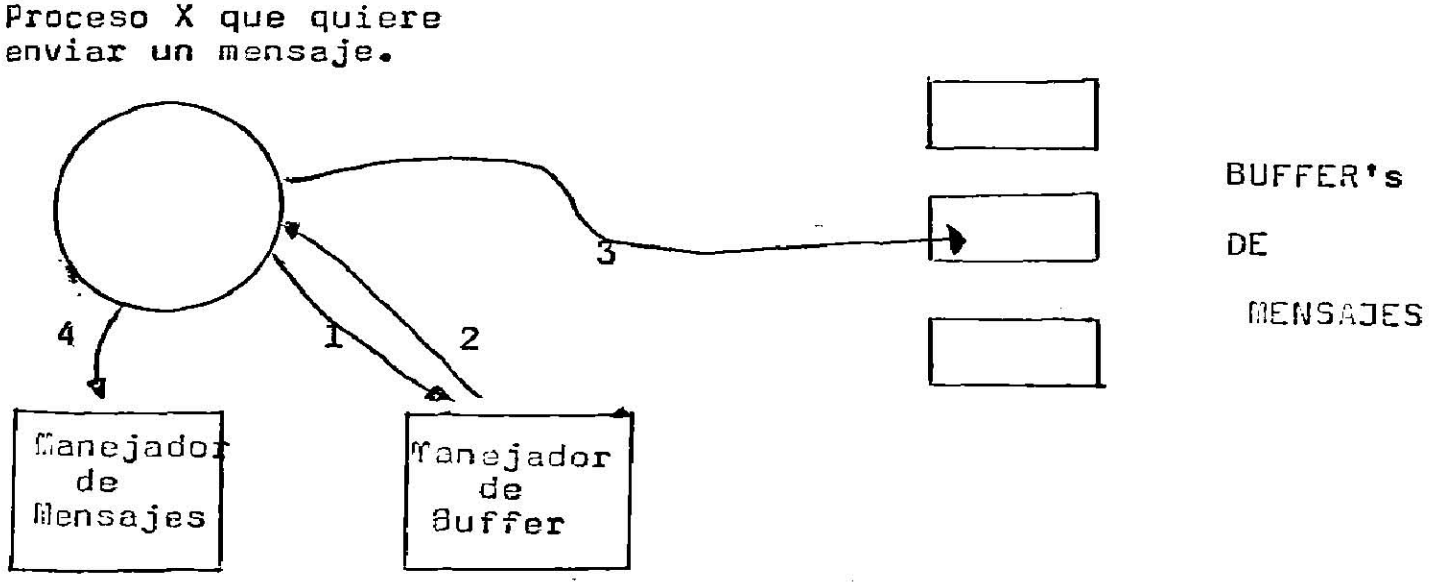
Controlar los periféricos de comunicación hombre-máquina.- Estos periféricos se manejan mediante los correspondientes "controladores" los cuales forman parte del sistema operativo. Todo el sistema que controla la entrada/salida hacia las memorias de masa y los periféricos de comunicación hombre-máquina, forman parte del SO.

Este mecanismo es suficiente para transmitir la mayoría de los mensajes del sistema. Existen ocasiones en que no es suficiente con los 40 bytes del texto para mandar toda la información. Para solucionar estas situaciones, el SO (concretamente el "manejador de buffer") dispone de todos los denominados "buffers de usuario" que poseen diversos tamaños (64 bytes - 2048 bytes). En este caso, cuando el proceso que desea enviar el mensaje prevee que no va a tener

suficiente con el buffer de mensaje, pide además que se le asocie uno de estos buffer de usuario indicando el tamaño que necesita. El SO se lo proporciona anotando en el buffer del mensaje el puntero del buffer de usuario. Este mecanismo solo está permitido para procesos ejecutándose en elementos de control tipo B.

Una vez que el proceso destinatario obtiene el buffer (de mensaje o de usuario) la información que necesita libera al buffer a fin de que pueda ser asociado a otro proceso que desee de nuevo enviar información. Este buffer liberado, queda de nuevo a disposición del manejador de buffers.

PASOS A DAR PREVIOS AL ENVIO DE UN MENSAJE



1) El proceso X que quiere enviar un mensaje pide que se le asocie un buffer de mensaje.
2) El manejador de buffer le pasa el puntero del buffer asociado.
3) El proceso rellena el buffer con la información adecuada.

4) El proceso pide la transmisión del TI y de los caminos de la red.

Cargadores e inicializadores.- Encargados de permitir la carga de los diferentes CE's por primera vez o como acción de mantenimiento, debido a un fallo.

En el CE de periféricos y mantenimiento, existen además otros módulos:

- Sistema de entrada/salida: encargado de controlar los periféricos de ordenador:
  - síncronos: cinta y disco.
  - asíncronos: pantallas, impresoras y TTY permitiendo así:
- la obtención de los GLS y DLS (almacenados en la memoria de masa de los periféricos síncronos), para cargar los en los diferentes CE de la central.
- el almacenamiento de información en las memorias de masa.
- la comunicación hombre-máquina a través de los periféricos asíncronos.

Control de Overlay.- Permite el uso de FMM's no residentes, controlando su carga y ejecución.

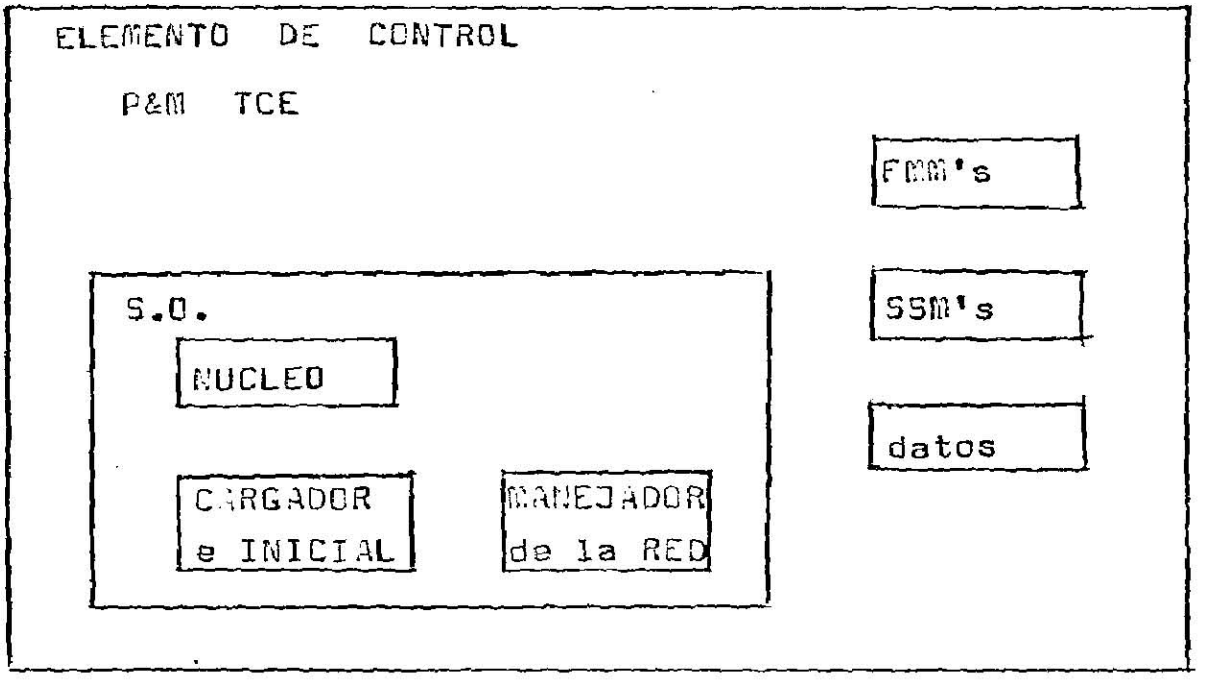
Núcleo del Sistema Operativo.

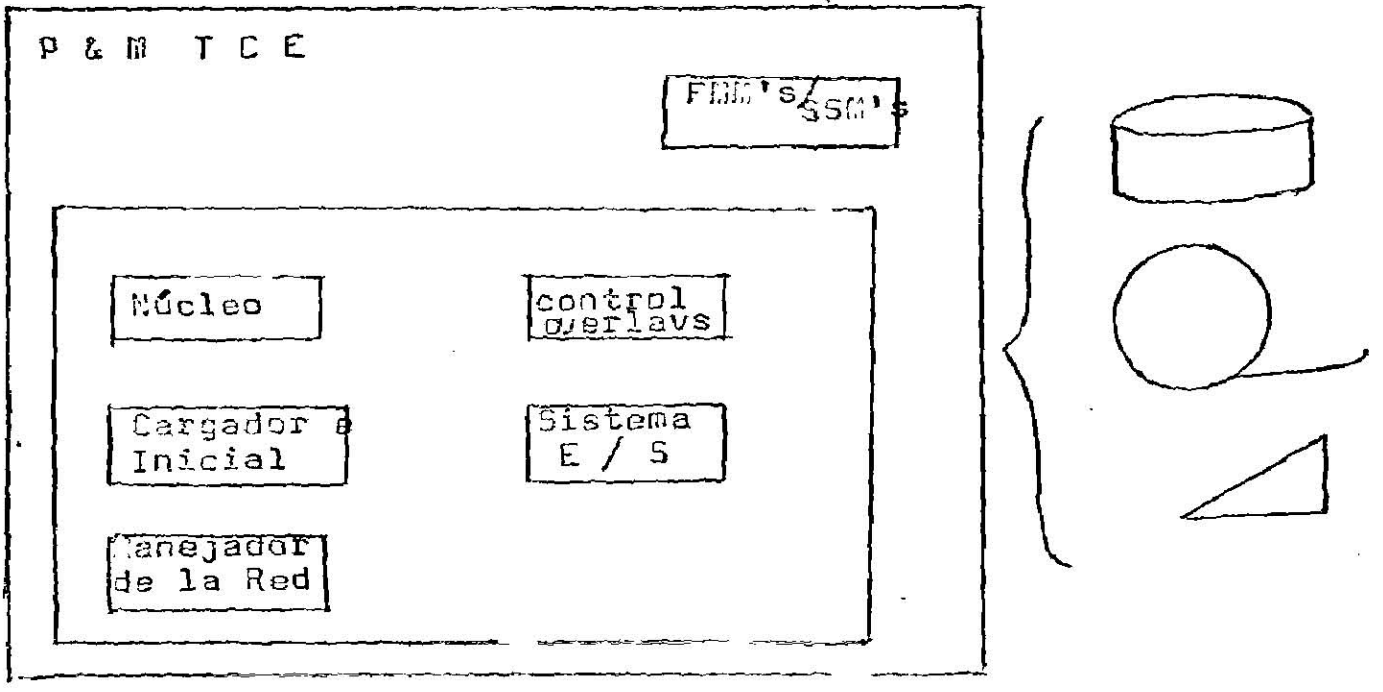
Parte del SO que se encarga de crear las condiciones para que se puedan ejecutar las diferentes FMM's y SSM's del sistema; no se ha diseñado como FMM ni como SSM.

Las funciones que realiza son las siguientes:

- Soportar la ejecución concurrente de muchos procesos corriendo en la misma máquina física. Para ello, "maneja

DIFERENTES COMPONENTES DEL SISTEMA OPERATIVO





procesos" (creándolos, activandolos y acabandclos) realiza la "planificación del tiempo de procesador" (indicando qué hacer en cada momento)

- Proporcionar a los procesos un mecanismo de comunicación, mediante mensajes.
- Proporcionar determinados servicios de tiempo.
- Proporcionar funciones soporte.
  - Manejo de buffers.
  - Manejo de errores.
  - etc.

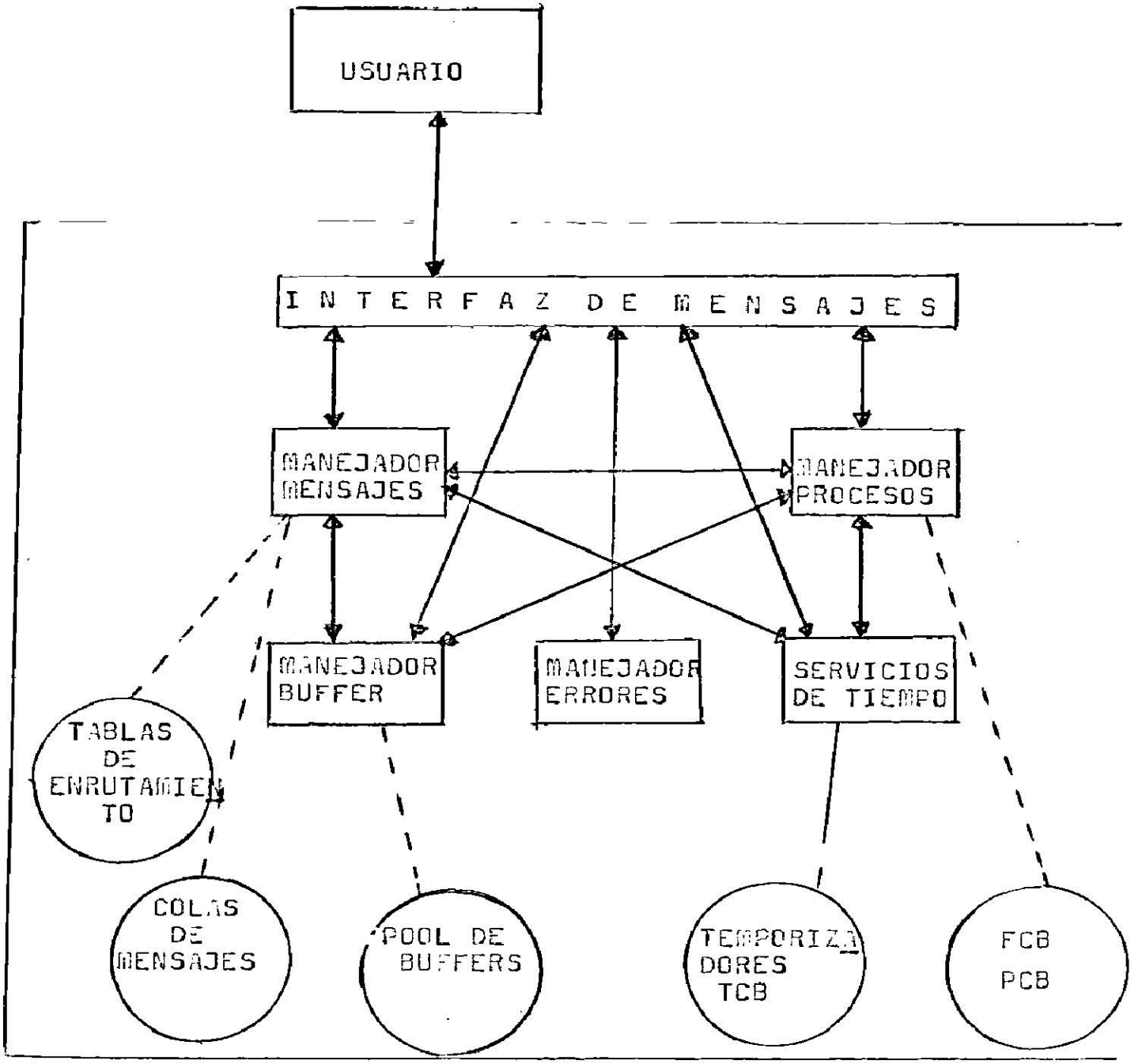
Manejador de Mensajes.

Se encarga de recibir los mensajes enviados por los procesos de las FMM's y analizarlos para determinar el destino físico de los mismos. Si como resultado del análisis sabe que el proceso destino está en ese procesador, almacena el mensaje hasta su posterior presentación. Si por el contrario el mensaje es para un proceso que se está ejecutando en otro procesador, le pasa dicho mensaje al Manejador de la Red indicándole la identidad física del microprocesador destino, para que este se encargue de su transmisión.

Cuando un proceso de una FMM necesita enviar un mensaje a otro proceso de otra FMM, escribe en un buffer la información a mandar y pide al SO su transmisión. Realmente esta petición es una llamada a esta parte del OSN, el cual va a analizar el destino del mensaje. Para ello, el manejador de mensajes dispone de unas tablas llamadas "Tablas de enrutamiento de Mensajes" (MRT), que le van a proporcionar parte de la información necesaria.

El tratamiento va a ser diferente, dependiendo de que se

ESTRUCTURA DEL NUCLEO



trate de mensajes básicos o dirigidos. Analicemos ambas situaciones:

a) Caso de Mensajes Básicos.

El proceso destino va a ser el supervisor de la FMM destino.

El SO tiene por cada FMM una zona de memoria donde almacena los datos necesarios de cada FMM. Esta zona va a jugar un papel en el enrutamiento. A esta zona se le denomina "Bloque de control de la FMM" (FMM Control Block, FCB)

La información almacenada en esta zona es:

- Información sobre el proceso supervisor.
- Número de procesos de aplicación que esa FMM puede tener.
- Dirección donde se encuentra almacenada. (CS, DS, SS)
- Dirección donde se encuentra el punto de la parte supervisora donde va a ser inicializada.
- Etc.

Cuando una FMM quiere enviar un mensaje básico lo hace de la forma:

"Transmitir-mensaje-básico X"

(TRANSMIT-BASIC-MSG X)

Cuando el Message Handler recibe el mensaje, por el nombre del mensaje, indexando con él en la tabla de enrutamiento obtiene:

- en caso de FMM almacenada en el mismo procesador, donde se encuentra el bloque de control de la FMM. Esta información es ya suficiente para conocer el proceso

destino. Ver figura página 84 .

- en caso de FMM almacenada en otro procesador, la identidad lógica del procesador destino. Con esta identidad lógica (y dado que lo que necesita es la física para poder pedir al "Manejador de la Red" (NH) el envío del mensaje, consultando en las ta--- blas de configuración, ver figura página 85 se obtiene la idéntidad física. Esta información es suficiente para pedir la transmisión del mensaje. Observar que una vez transmitido al otro procesador a través del NH el cual lo presentará a su manejador de mensajes, que repetirá al analisis obteniendo en este caso la identidad del FCB (pues ya se trata de una FMM interna a ese CE).

Hasta ahora hemos supuesto que con el nombre del mensaje básico el "Manejador de Mensaje" tiene sufuciente información para conocer el destino. Esto, a veces es verdad pero no en todos los casos. Veamos algunas situaciones diferentes.

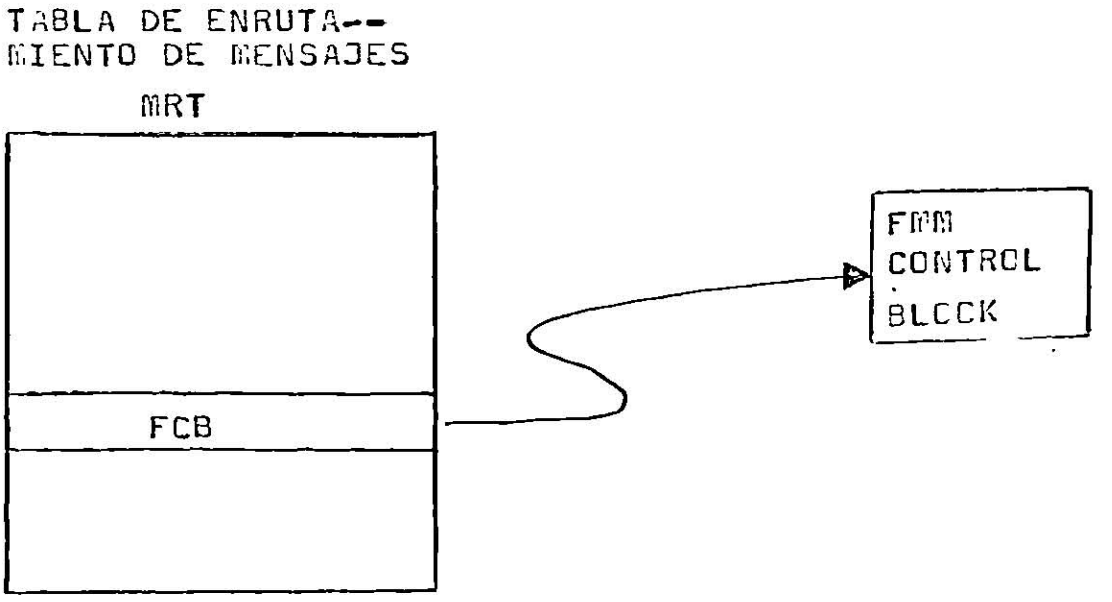
En el módulo de P&M se encuentran los periféricos clásicos y por tanto en P&M TCE están los manejadores de dispositivos (DH) de estos periféricos. Como se vió en el apartado de máquinas virtuales, a estos manejadores se les dá una orden abstracta, por ejemplo "escribir" y estos la concretizan sobre los correspondientes dispositivos.

Los mensajes básicos que estos DH pueden escribir llevarán el mismo nombre, necesitándolo el "manejador de mensajes"

ENRUTAMIENTO DE MENSAJES BASICOS

(caso de mensaje interno)

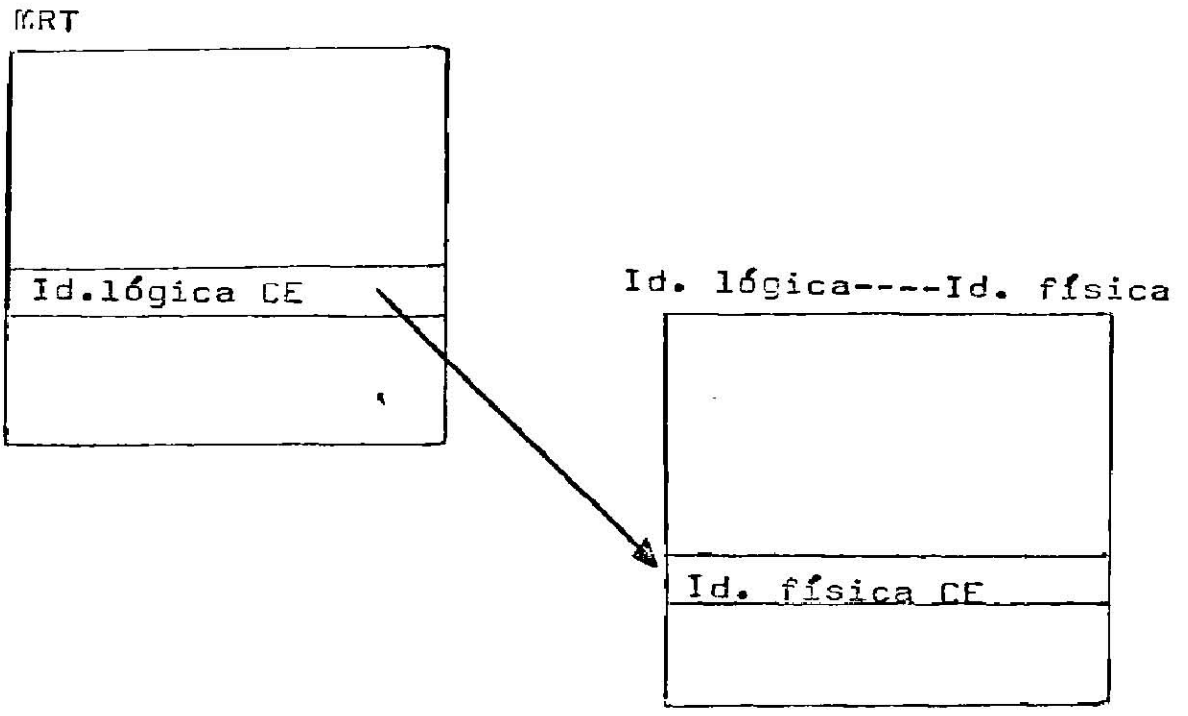
TRANSMIT - BASIC - MSG X (P1...PN)



ENRUTAMIENTO DE MENSAJES BASICOS

(caso mensaje externo)

TRANSMIT - BASIC - MSG X (P1...PN)



una información adicional para saber a cual de los DH va dirigido. Esta información se la proporciona el proceso que envía el mensaje pues en este caso la petición de envío es del estilo:

"Transmitir-mensaje-básico ESCRIBIR para discriminador"
(TRANSMIT-BASIC-MSG X FOR discri.)

En este caso para el enrutamiento es necesario, además de la tabla de "enrutamiento" tal como aparece en la figura de la página 87 .

Una situación diferente es la siguiente: consideremos un mensaje que debe llegar a una FMM almacenada en un ACE de líneas, todos con un conjunto de programas idénticos o similares. Es lógico pensar que al enviar un mensaje a la FMM de un micro específico se deberá indicar a qué microprocesador (de los muchos que contiene esa FMM) va dirigido. En este caso la petición se hará de la forma:

"Transmitir-mensaje-básico X al (CE-lógico)"
(TRANSMIT-BASIC-MSG X INTO (CE-lógico)

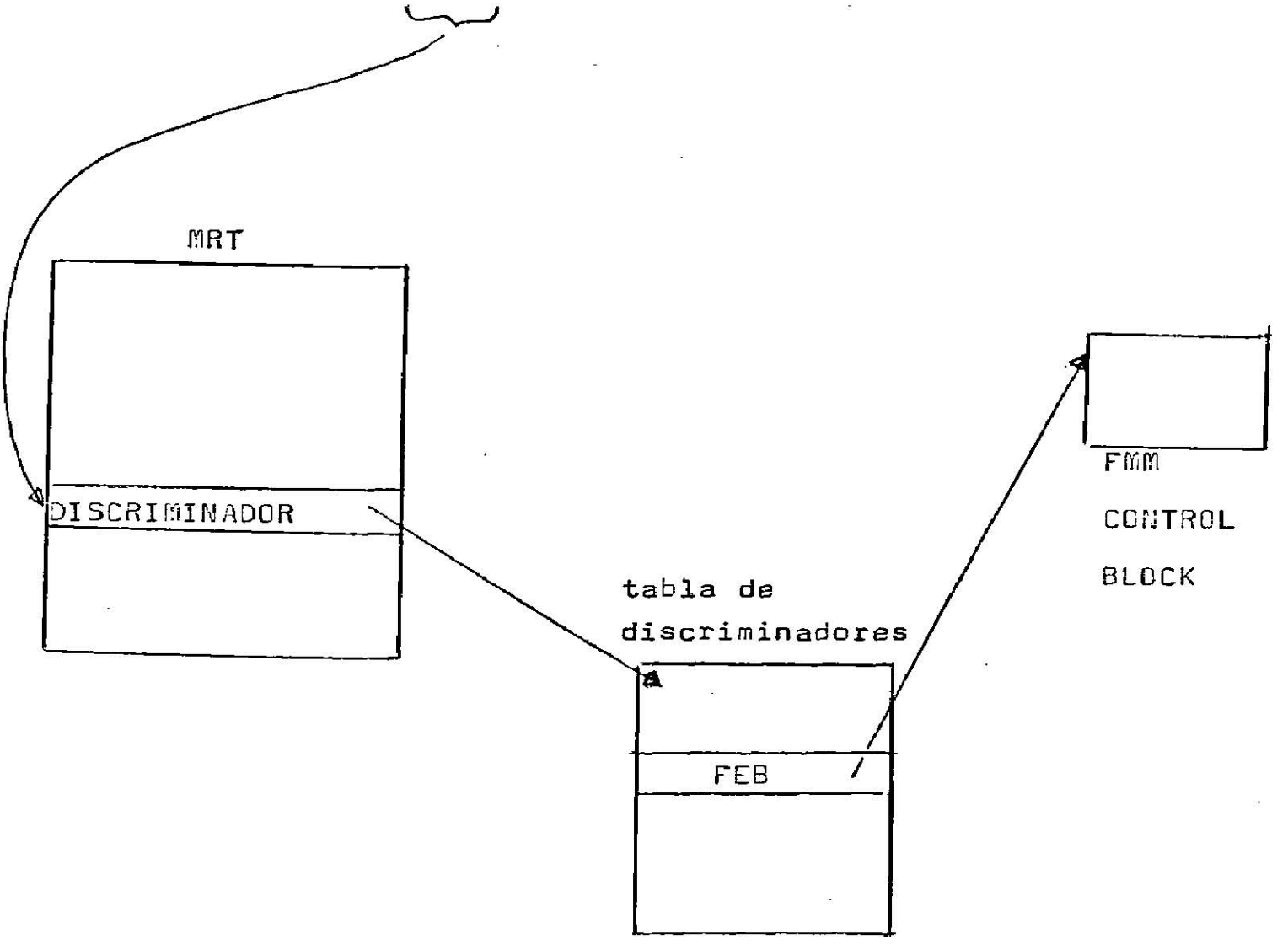
En este caso, para calcular el destino físico, el Manejador de mensajes no necesita mirar la tabla de enrutamiento sino simplemente los "datos de configuración" que transforman las identidades lógicas en físicas, tal como aparece en la figura de la página 88 .

Así pues, si 2 FMM's del mismo procesador admiten el mismo mensaje básico, al enviar el mensaje se deberá indicar el discrininador (FOR - Discriminador). Si 2 FMM's almacenadas en distinto microprocesador reciben el mismo

ENRUTAMIENTO DE MENSAJES BASICOS

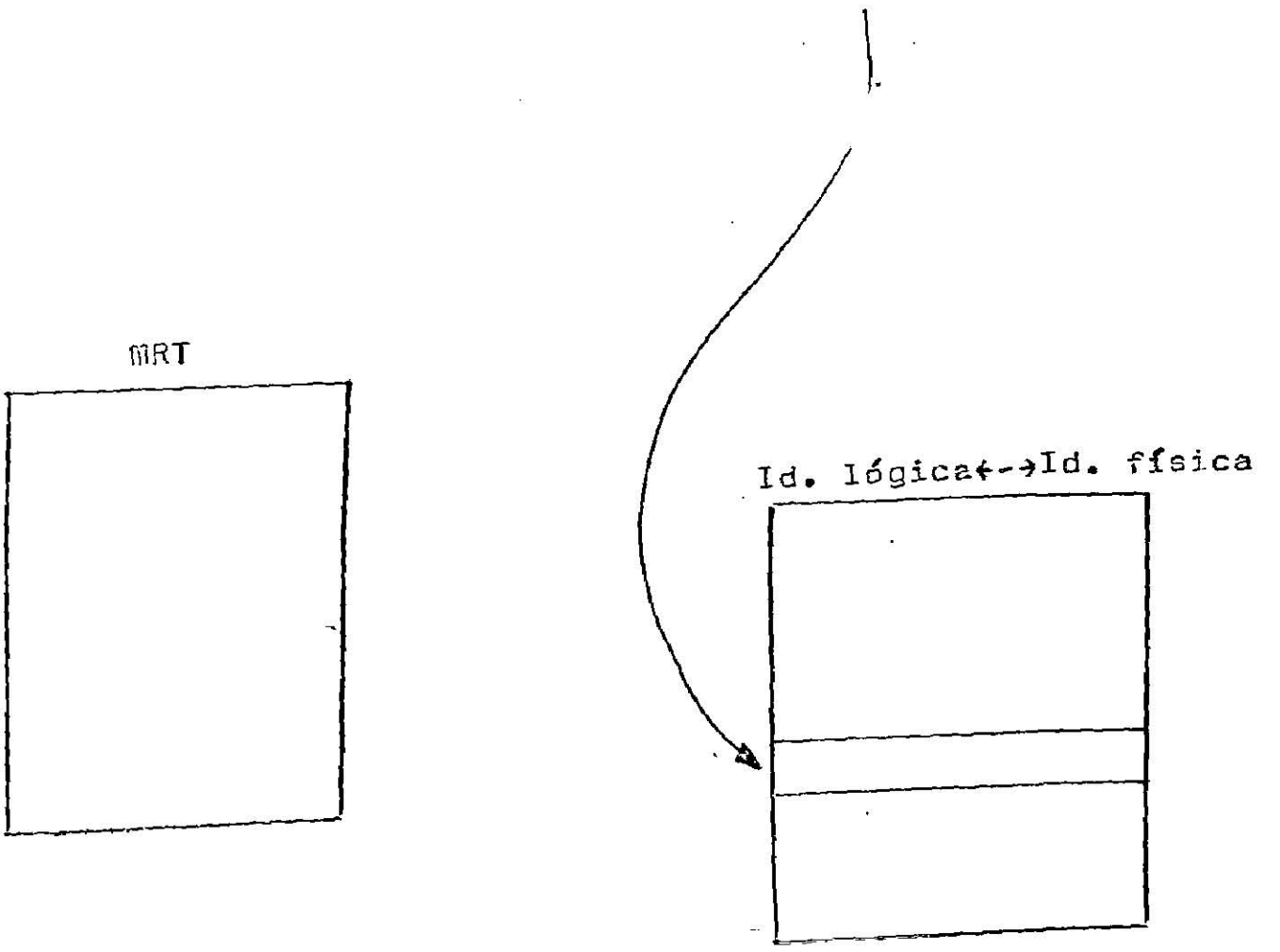
(con discriminador)

TRANSMIT - BASIC - MSG X (PL...PN) FOR (discriminador)



ENRUTAMIENTO DE MENSAJES BASICOS

TRANSMIT-BASIC-MSG X (P1...PN) INTO (Logical-CE)



mensaje básico, al enviarlo se debe indicar el micro destino (INTO - id. lógico del CE). La situación combinada (FOR - INTO) también es posible.

Otra posibilidad es que el proceso enviante indique, en lugar de la identidad lógica del CE destino, la identidad física. En este caso, el "Manejador de Mensajes" ya no necesita consultar en ninguna tabla sino que le pasa al "Manejador de la red" directamente la información. Este tipo de transmisión se pide de la forma: TRANSMIT-BASIC-MSG X ONTO (CE-FISICO). Finalmente, otra información posible a añadir en la petición de envío de un mensaje básico, es la pedir que se establezca un camino en la red e indicarle al OSN que le envíe ese mensaje básico a través de ese camino. Cuando se hable de "caminos de usuario", la petición será de la forma: "Transmitir-mensaje-básico X a través-de- (camino) "(TRANSMIT-BASIC-MSG X VIA (CAMINO)".

En este caso el manejador de mensajes lo pasa directamente al "manejador de la red"el camino por el que debe enviar el mensaje.

b) Caso de Mensajes Dirigidos.

Como su nombre indica, estos mensajes van siempre dirigidos a un proceso en particular (que puede ser supervisor o de aplicación). En este caso, el que envía el mensaje debe indicar la identidad del proceso destino. En la identidad de un proceso viene información sobre en qué CE se está ejecutando el proceso, por tanto el

enrutamiento por parte del "Manejador de Mensaje" es directo. La petición se hace de la forma:

"Transmitir-mensaje-dirigido X al (proceso)"

(TRANSMIT-DIRECTED-MSG X TO id-proceso) en el caso de mensajes dirigidos, al igual que en los básicos, también se puede pedir la transmisión a través de un camino en particular.

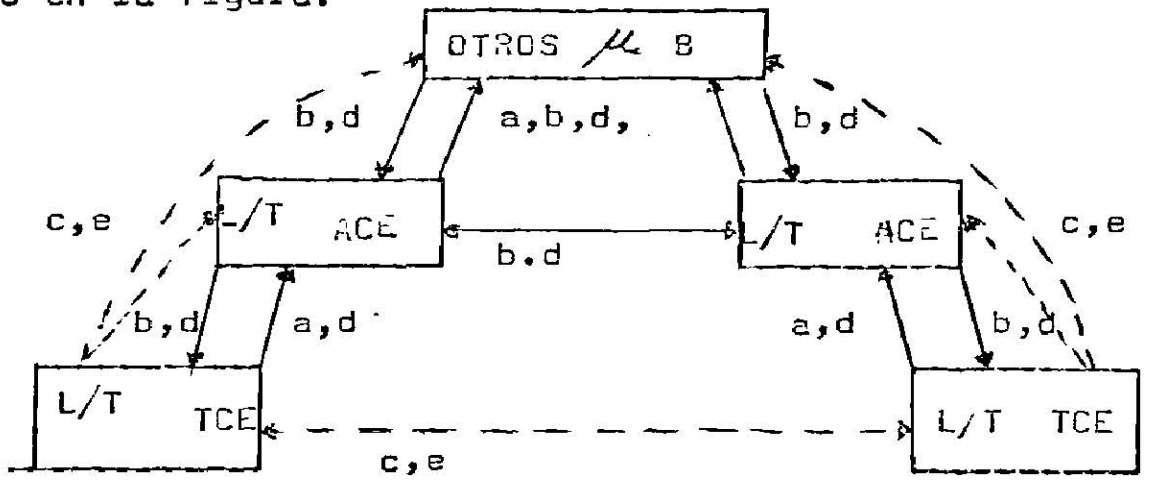
(TRANSMIT-DIRECTED-MSG X VIA (camino)

RESUMEN de las diferentes formas de pedir la TRANSMISION DE MENSAJES BASICOS

TRANSMIT-BASIC-MSG	nombre	(P1---Px)		
TRANSMIT-BASIC-MSG	nombre	(P1---Px)	FOR	(discri.);
TRANSMIT-BASIC-MSG	nombre	(P1---Px)	INTO	(CE lógico);
TRANSMIT-BASIC-MSG	nombre	(P1---Px)	INTO FOR	(CE lógico) (discri.);
TRANSMIT-BASIC-MSG	nombre	(P1---Px)	ONTO	(CE físico);
TRANSMIT-BASIC-MSG	nombre	(P1---Px)	ONTO FOR	(CE físico) (discri.);
TRANSMIT-BASIC-MSG	nombre	(P1---Px)	VIA	(camino);
TRANSMIT-BASIC-MSG	nombre	(P1---Px)	VIA FOR	(camino) (discri)

Restricciones de enrutamiento

Existen restricciones de enrutamiento entre las FCM's almacenados en los diferentes CE's de la central, que aparecen representadas en la figura:



a) BASIC nombre FOR (discriminador)

b) BASIC nombre FOR (discriminador)

INTO (identidad-lógica-del-CE)

ONTO (identidad-física-del-CE)

c) BASIC nombre FOR (discriminador)

VIA (id. camino)

d) DIRECTED nombre TO (id. proceso)

e) DIRECTED nombre VIA (id. camino)

Una vez que se conoce el destino físico del mensaje, el Manejador de Mensajes realiza.

- Si el mensaje es para otro procesador, le pide al NH que lo envíe a través de la red.

- Si el mensaje es interno, lo almacena.

Para almacenarlo dispone de unas zonas con la estructura de cola. Cada mensaje, en función de lo urgente que sea la tarea que va a arrancar, tiene asignada una prioridad, existiendo hasta 8 prioridades diferentes. Debido a ello, existen hasta 8 colas de mensajes diferentes (uno por prioridad) donde el "Manejador de Mensajes" almacena éstos hasta que llega el momento de presentárselo al proceso.

Resumiendo la actuación del "Manejador de Mensajes", tenemos:

- Aceptar las peticiones de envío de mensajes desde las FMM's.
- Averiguar la identidad física del micro destino.
- En caso de ser un micro diferente, pedir al "Manejador de Red" la transmisión del mensaje.
- En caso de mensaje interno al micro, almacenarlo en la cola adecuada, dependiendo de la prioridad hasta su posterior presentación.

Manejador de Procesos.

Parte del OSN que se encarga del control de los procesos: activándolos en el momento adecuado (para lo cual se encarga de distribuir el tiempo de procesador), creándolos y destruyéndolos.

Analicemos primeramente cómo se distribuye el tiempo del procesador (Schedulling). Las cosas que deben ejecutarse en un pro-

cesador son las siguientes:

- Procesos de las diferentes FMC's.
- Rutinas de las SSM's
  - Periódicas.
  - Interrupción de dispositivo.
  - Manejador de eventos.
  - Interface.

El sistema operativo las irá dando entrada sucesivamente de acuerdo con unas prioridades establecidas, englobándolas en 3 grandes bloques obtenemos:

- Rutinas asociadas a interrupciones (de reloj y de dispositivos)
- Manejadores de eventos.
- Procesos.

Observar que las rutinas periódicas se han englobado en el primer grupo pues realmente entran en ejecución debido a que ha llegado una interrupción de reloj, y las de interface no se han incluído, debido a que no son arrancadas por el S.O. sino por los procesos.

Las rutinas asociadas a interrupciones corren siempre con las interrupciones inhibidas y son las más prioritarias. Esto significa que si llega por ejemplo una interrupción de dispositivo (p.ej. del disco), comienza a tratarse, aunque en ese momento llegue la interrupción de reloj por el Hw, si el Sw no la acepta y queda esperando a que se permitan de nuevo las interrupciones.

Si se presentan si ultánea mente la de reloj y la de periféricos se considera más importante la de reloj comenzándose antes

su tratamiento.

En la figura de la página 95 se ha representado este esquema de prioridades.

A todo microprocesador del sistema, además de la interrupción de reloj (y de periféricos para P&M TCE) le llega una interrupción que no se puede enmascarar representando determinadas situaciones de error. Al no poderse enmascarar, cuando se presenten esas situaciones de error inmediatamente se comienza su tratamiento a fin de que el micro detecte y elimine el error lo antes posible.

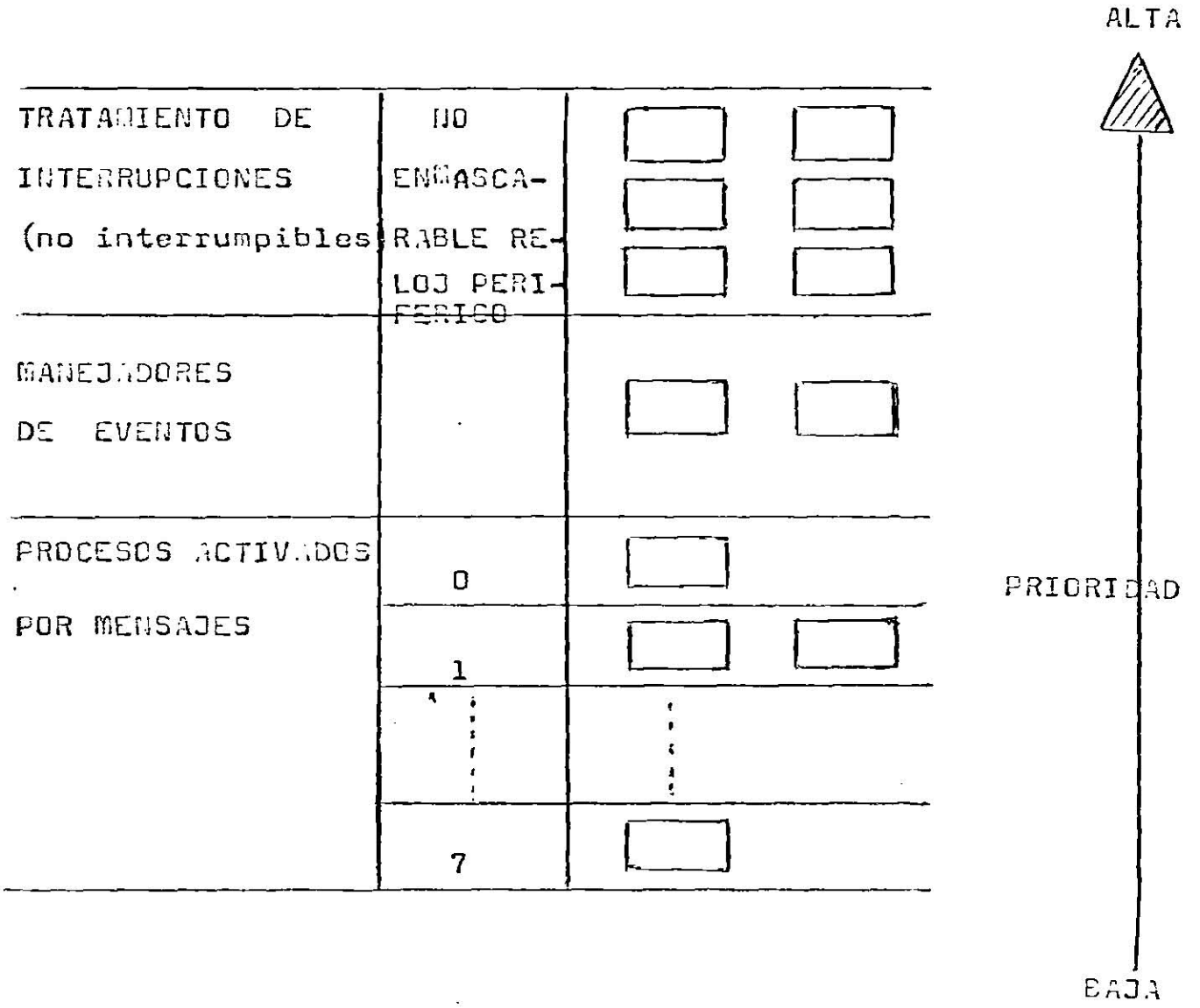
Siguiendo en el orden de prioridades, el siguiente bloque que encontramos es el de los manejadores de eventos. Estos entrarán cuando algún problema ejecutado previamente (bien rutinas de SSM's o procesos haya pedido su entrada, posicionando adecuadamente los flags de eventos.

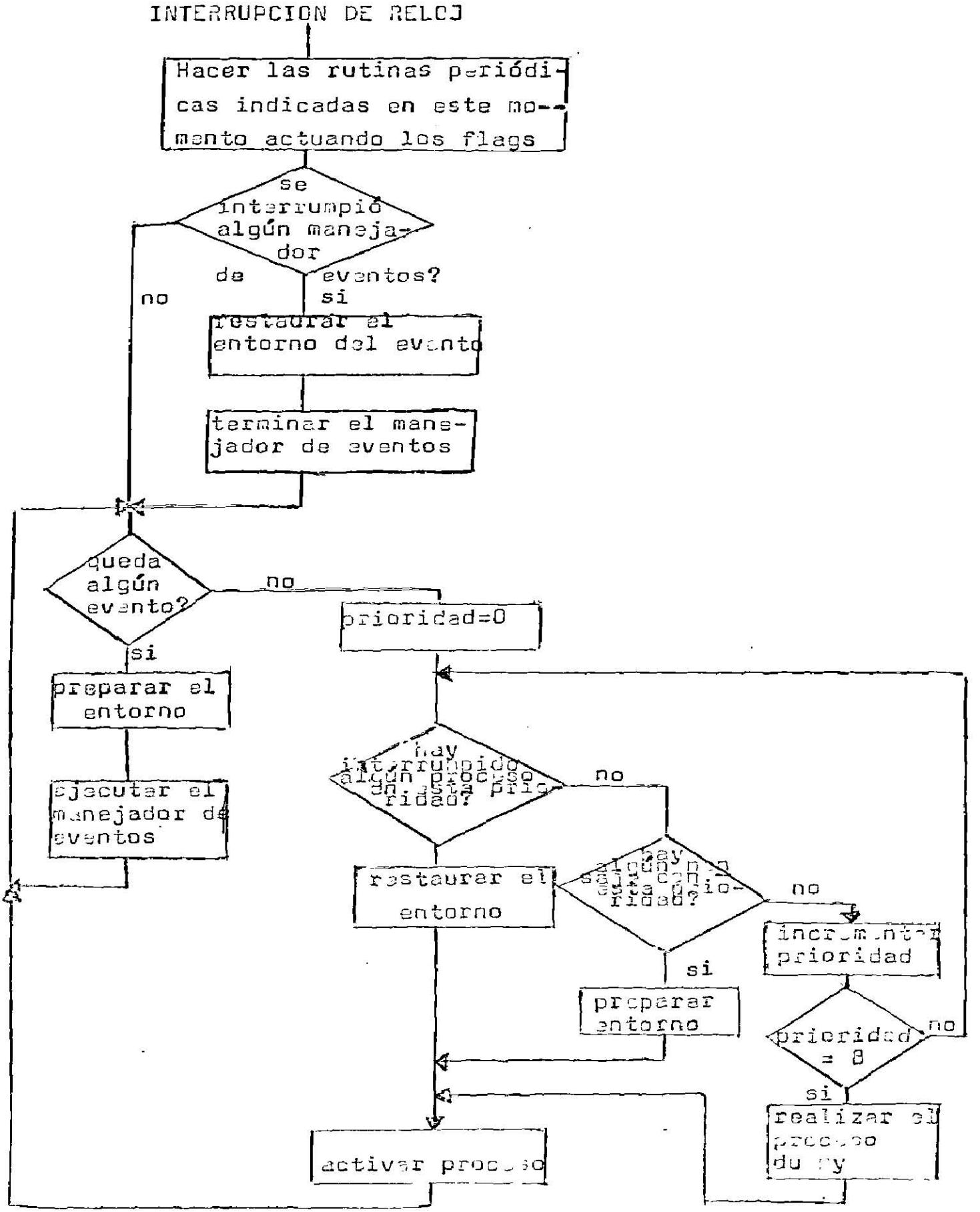
Finalmente, cuando no quede ningún manejador de eventos al que dar entrada, se pase a activar procesos pasándole mensajes. Para ello se va extrayendo de la cola de mensajes, empezando por la prioridad cero. En la figura de la página 96 aparece el algoritmo de Schedulli.

Como se apreciará en la figura, no se pasa a activar los procesos hasta que la pregunta ¿queda algún evento? se sale por la rama del no, comenzando por la prioridad cero.

Observar que una vez que se acaba la ejecución de un proceso previamente activado, se vuelve a preguntar si hay eventos, para activar de nuevo los correspondientes manejadores, volviendo de nuevo a comenzar por la cola más prioritaria.

NIVELES DE PRIORIDAD EN LA EJECUCION




INTERRUPCION DE RELOJ
Hacer las rutinas periódicas indicadas en este momento actuando los flags
se interrumpió algún manejador de eventos?
no
si
restaurar el entorno del evento
terminar el manejador de eventos
queda algún evento?
no
si
preparar el entorno
ejecutar el manejador de eventos
prioridad=0
hay interrumpido algún proceso en esta prioridad?
no
si
restaurar el entorno
preparar entorno
incrementar prioridad
prioridad = 8
no
si
activar proceso

Cuando ya no quedan mensajes en la cola de una determinada prioridad (salida del "no" a la pregunta "quedan mensajes con esta prioridad") se incrementa ésta y se comienza analizando si fue interrumpido un proceso tratando un mensaje con esa prioridad. Esto significa lo siguiente: imaginémos que cuando se estaba ejecutando un proceso tratando un mensaje con prioridad 3, llega la interrupción de reloj y se dá entrada a las rutinas periódicas asociadas a esa interrupción que dando ese proceso momentáneamente interrumpido. ¿Cuándo se reanudará dicho proceso?.

Analizando el algoritmo se puede contestar que será cuando no quede ningún evento pendiente y las colas de mensajes de prioridad 0, 1 y 2 estén vacías.

El proceso "dummy" se ejecuta cuando ya no tiene el micro nada que hacer y consiste simplemente en un proceso que se envía mensajes a sí mismo.

Siguiendo con con el estudio del Gestor de procesos, éste se encargará de, cuando no hay interrupción, mirar los flags de eventos para activar sus manejados y finalmente pedir al "Manejador de mensajes" que le indique , cuál es el proceso correspondiente al primer mensaje anotado en la cola más prioritaria. Cuando el Manejador de mensajes se lo indique, preparará el entorno del proceso, es decir cargará en los registros del ordenador la información que tenía cuando el proceso se paró (se quedó en estado de espera) y devolvió el control al S.O. Una vez que el Gestor de Procesos ha restaurado el entorno, el Manejador de mensajes le pasa ya el mensaje, arrancando la ejecución de éste.

Otra función del "Gestor de Procesos" es la de crear y

terminar los procesos. Analicemos detenidamente estas funciones.

Creación de Procesos.

Como se ha indicado anteriormente, es la parte supervisora de una FMM la que va a decidir en un momento dado, crear un proceso de su parte de aplicación, pidiéndole al Gestor de Procesos su creación. Esto se hace sencillamente llamando a la procedure CREATE-PROCESS. Cuando recibe esta petición, el Gestor de procesos la realiza en la siguiente forma:

El Gestor de Procesos tiene, por cada proceso que se está ejecutando en ese CE, una zona de memoria donde guarda toda la información que necesita saber de ese proceso. A esta zona se le denomina "bloque de control del proceso" (process control block, PCB) la información que contiene esta zona es la siguiente:

- ESTADO en que se encuentra el proceso. Los diferentes estados en que se puede encontrar son: ejecutándose, esperando, interrumpido, libre (no asignada la zona, en ese momento, a ningún proceso)
- PRIORIDAD del mensaje con el que esté trabajando.
- PLUS de aparición.
- IDENTIDAD de la FMM al que pertenece el proceso.
- PUNTERO al buffer del mensaje con el que está trabajando.
- INDICACION de si el proceso tiene abierto un temporizador.

En un CE existirán tantos PCB como número máximo de procesos simultáneos se espera que haya en el micro. En un momento de-

terminado un PCB puede estar libre asignado a un proceso.

Cuando se le pide al Gestor de Procesos la creación de uno, busca un PCB libre y se lo asigna, anotando en él la información necesaria. La identidad del proceso consta de:

- Identidad del CE.
- Plus.
- No. de proceso.

realmente en este tercer campo lo que aparece es el No. del PCB asignado a ese proceso.

Una vez creado el proceso de aplicación, los mensajes que le llegan son siempre mensajes dirigidos (recordar que los mensajes básicos no llegan a los procesos de aplicación). Cuando un proceso envía un mensaje dirigido a otro proceso, en la petición indica la identidad del proceso destino (TRANSMIT-DIRECTED-MSG TO (id-proceso-destino) indicando así, cuál es el PCB del mismo. Mirando en el PCB ya obtiene el Gestor de Proceso la información necesaria.

Terminación de Procesos.

Existen diferentes formas de acabar un proceso:

- Por petición del propio proceso: cuando el proceso detecta que ya ha acabado de realizar la función para la que fué creado, pide al S.O. que lo termine, indicándolo con la primitiva "TERMINATE".
- Por petición del proceso supervisor: en un momento determinado éste puede pedirle al S.O. la terminación de todos los procesos de su parte de aplicación.

- Por decisión del Gestor de Procesos, debido a un fallo (terminación forzada).

Veámos algún ejemplo, que obligue a una toma de decisión de este tipo. Consideremos la situación en que un proceso ha quedado en un WAIT esperando que otro proceso (que en principio podemos suponer en otro procesador diferente) le envíe un mensaje. Imaginémos que ese procesador cae y que el proceso se pierde; lógicamente el proceso que está esperando puede quedarse eternamente en ese estado de espera. Para evitar esta situación, el S.O. va realizando un control sobre los diferentes procesos temporizando sus tiempos de espera. Si al cabo de un determinado tiempo un proceso no recibe ningún mensaje, se presupone que es debido a algún fallo y se aborta el proceso.

Hay que indicar que hay estados de espera en los que el proceso puede estar eternamente en ellos sin que signifique que haya habido un error. P.ej: un proceso que controla un dispositivo puede estar infinitamente en "libre" cambiando dicho estado cuando el dispositivo sea "tomado" para realizar algún trabajo. Para diferenciar estos estados de espera (en los que el proceso puede estar en ellos eternamente) de los vistos anteriormente (en que debe existir un control de temporización por parte del S O), el diseñador de la FSM debe indicar que se trata de un PEST (POTENTIALLY ETERNAL STATE).

Sea cual sea la causa de terminar el proceso, el Gestor del proceso:

- Libera la zona de memoria de los datos dinámicos.
- Libera el PCB quedando libre para asignárselo a un

proceso que se cree posteriormente (habrá, lógicamente, que incrementar el plus, -).

Facilidades de Tiempo.

El S.O. va a proporcionar a las diferentes FMP's/SSM's del sistema, una serie de facilidades de temporización, siendo el módulo "servicios de tiempo" (Time Service) el encargado de proporcionarlas. Veamos cual van a ser esas facilidades:

Un proceso puede pedir al sistema operativo que lo active:

- En un momento determinado del día (temporización absoluta).

  p. ej.: "despiertame a las 12".

  se indica las horas y los minutos con un rango de 24 horas. El S.O. "despertará" con una precisión de un minuto.

- Al cabo de un determinado tiempo (temporización relativa).

  p. ej.: "despiértame dentro de 25 seg.".

  se indica los segundos y décimas de segundo en un rango de una hora (3599.9 seg.). El S.O. despierta con una precisión de 1 décima de segundo.

- Periódicamente (con el período que quiera) bien a partir de un momento del día o al cabo de determinado tiempo (temporización periódica absoluta y temporización periódica relativa).

  p.ej.: absoluta: "a partir de las doce, cada 5 minutos".

relativa: "a partir de 5 seg. cada segundo"

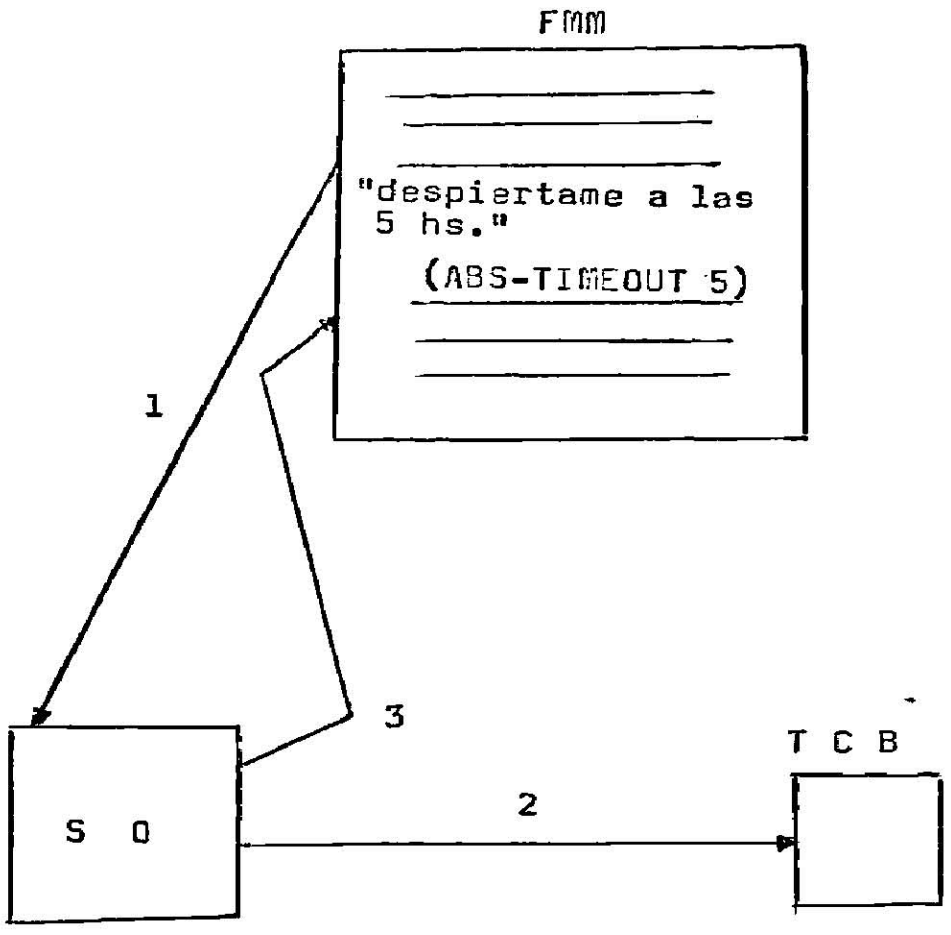
Cuando un proceso realiza una petición de este tipo al S. O., el módulo "Servicios de tiempo" le abre un temporizador (timer control block, TCB) y se lo asocia al proceso (figura de la página 103). Al expirar el temporizador le manda un mensaje al proceso indicándoselo. La entrega del mensaje activa el proceso. El proceso que tiene abierto y asociado un temporizador (cuando se habló del PCB, bloque de control del proceso, se indicó que también tiene éste información sobre si tiene o no asociado un temporizador, o lo que es lo mismo, tiene indicación del TCB asociado) puede, si le interesa, indicar al S.O. que lo pare pues ya no le interesa ser despertado. Gracias a este mecanismo, los procesos disponen de un servicio de temporización muy fácil de manejar y bastante potente.

Otra misión de este módulo del núcleo del sistema operativo "servicio de tiempo" es llevar la cuenta de las interrupciones de reloj que le llegan al CE para, en función de las que han llegado, hacer llegar las correspondientes rutinas periódicas.

P. ej.: supongamos que una rutina periódica se tiene que ejecutar cada 100 minutos. Como el reloj interrumpe cada 5 msg. cada 20 interrupciones de reloj debe entrar la rutina.

Por el puerto 5 cada TI llega, procedente del módulo de reloj y tonos, la hora del día (Time of Day, TOD) expresada en horas, minutos, segundos y décimas de segundo. Este módulo "servicios de tiempo" se encarga de leer de este puerto esta información para podérsela proporcionar a cualquier FMM que la necesite (p. ej. para emitir informe indicando el momento en que produce, etc.)

USO DE LOS TEMPORIZADORES



(1) Petición al S.O. de un temporizador.

(2) El S.O. busca un temporizador y le pone el valor.

(3) Devuelve al proceso la id. del temporizador que ha abierto.

(4) Cuando expire el temporizador, el S.O. envía al proceso un mensaje, que activa al proceso, indicándole que ha expirado el temporizador.

Otra función del módulo es la de actualizar el "sanity timer". El Hw inherente al procesador tiene un temporizador que debe ser actualizado periódicamente, pues de lo contrario provoca una interrupción no enmascarable que indica al micro que algo está fallando pues no ha sido actualizado. A este temporizador se le denomina "sanity timer" y es este módulo del S.O. "servicios de tiempo" el que se encarga de su actualización.

Manejo de Buffers (Buffer Manager)

Siguiendo con nuestro estudio del núcleo del S.O. de la figura de la página 61, encontramos el módulo "Manejador de Buffer" cuya función es centralizar y controlar los recursos de memoria necesitados por las FMM's para transmitirse información. Hasta ahora cuando hemos hablado de transmisión de mensajes de un proceso a otro mediante la intervención del "Manejador de Mensajes", lo hemos hecho de forma global sin analizar en qué consiste realmente la transmisión.

El S.O. dispone para este fin de un conjunto de zonas de memoria con estructura de buffer que va a utilizar para la comunicación de los procesos. Estos buffers, denominados buffer de mensaje tienen un tamaño fijo de 64 bytes y consisten en 2 partes: cabecera y texto. La cabecera contiene el número del mensaje (lo que hasta ahora hemos venido llamando "nombre" del mensaje), el tamaño del texto, la prioridad del mensaje e información sobre el enrutamiento (indicación de si es "básico", "básico FOR - discriminador", "básico INTO elemento-control", etc.). El texto incluye información que el proceso enviante quiere hacer llegar al proceso destino. El

tamaño del texto es, como máximo de 40 bytes.

Este mecanismo es suficiente para transmitir la mayoría de los mensajes del sistema. Existen situaciones en que no es suficiente con los 40 bytes del texto para mandar toda la información para solucionar estas situaciones, el S.O. (concretamente el "MANEJADOR DE BUFFER") dispone de los denominados "buffers de usuario" que posee diversos tamaños (64 bytes-2048 bytes). En este caso, cuando el proceso que desea enviar el mensaje prevee que no va a tener sufuciente con el buffer de mensaje pide además que se le asocie unos de estos buffer de usuario indicando el tamaño que necesita. El S.O. se lo proporciona en el buffer del mensaje el puntero del buffer de usuario. Este mecanismo solo está permitido para procesos ejecutándose en elementos de control diferentes a L/T TCE (recordar que ya se indicó que el S.O. de los micros de la línea y troncales es menos potente).

Una vez que el proceso destinatario obtiene del buffer (de mensaje o de usuario) la información que necesita, libera el buffer a fin de que pueda ser asociado a otro proceso que desee de nuevo enviar información. Este buffer liberado, queda de nuevo a disposición del "Manejador de buffers".

Manejador_de_Errores.

Aunque este módulo se encuentra formando parte del núcleo del S.O. de los diferentes CE's del sistema, desde el punto de vista funcional forma parte del subsistema de mantenimiento y se analizará su función detenidamente.

Su misión va a ser la de recopilar todos los errores que detecten en ese CE los diferentes procesos que se ejecutan en el mismo. Ejemplos de estos errores detectados pueden ser:

- Errores en el Hw detectados por el correspondiente "Manejador de dispositivo" que trata ese Hw.
- Un parámetro de una subrutina que ha sido declarado de un determinado rango y se ha salido de él.
- La cola de envío de mensajes está llena indicándose que hay algún fallo pues se ha llegado a la saturación.

Cuando un proceso, una rutina de una SSM o incluso el S.C. detecta algún error se lo indica al "Manejador de errores" el cual se encarga de:

- Informar a una parte centralizada de mantenimiento.
- Intentar una recuperación local para, si puede, ser del tipo: abortar el proceso que está teniendo errores, inicializar el micro, etc...

BASE DE DATOS

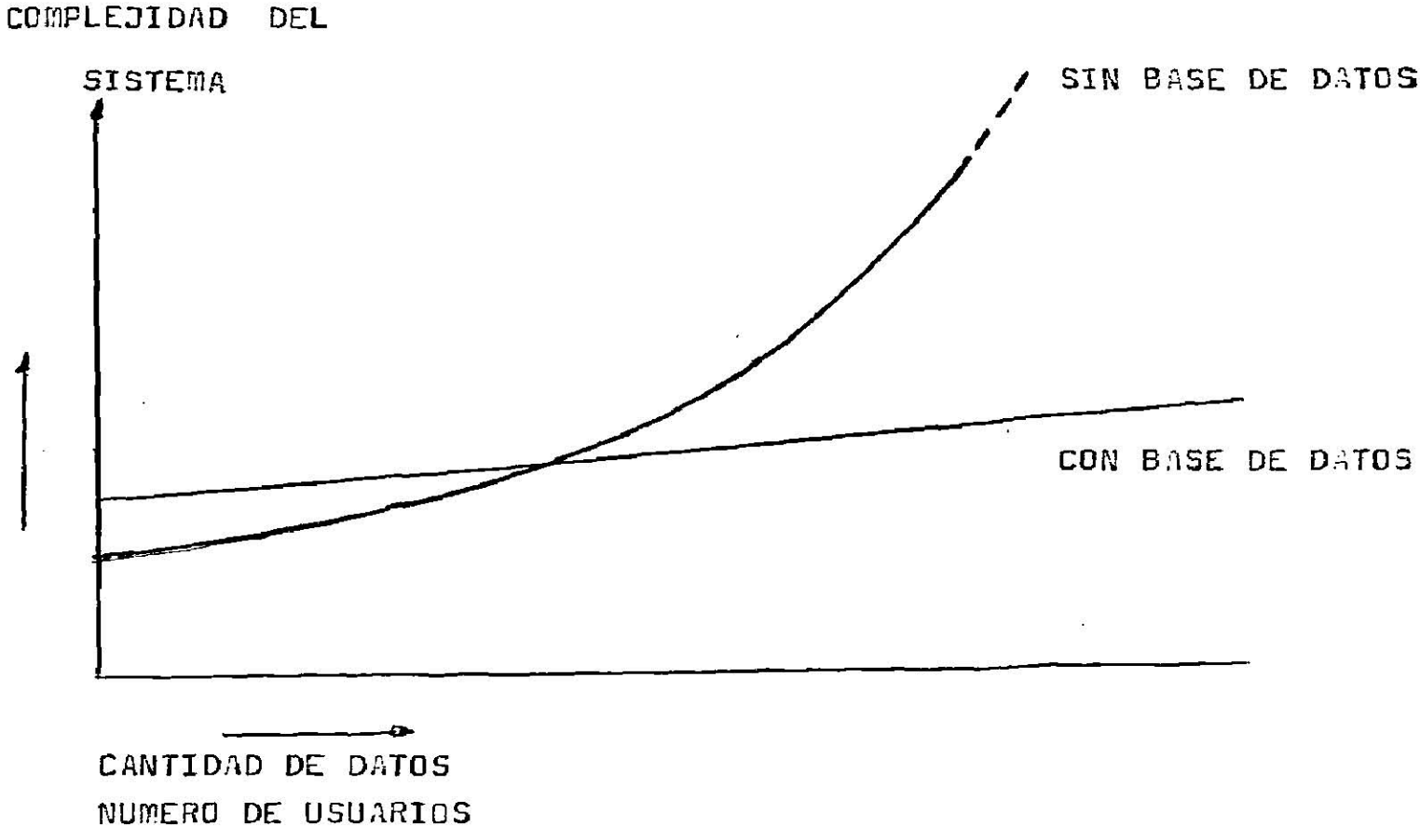
Introducción.

La mayoría de los datos estables de la central (entendiendo por estables aquellos que tienen memoria asignada durante toda la vida de la central, aunque puedan variar sus valores) se han asociado constituyendo una base de datos.

¿Cómo se puede justificar la necesidad de una base? En el sistema 12 40 van a existir una gran cantidad de datos estables así como muchas FMM's que necesiten trabajar con los datos (usuarios de los datos), de tal forma que un mismo dato va a ser manejado por diferentes usuarios perteneciendo incluso a diferentes subsistemas. Intentando aplicar el concepto de máquina virtual, se ha visto la necesidad de aislar a los diferentes usuarios de la problemática de los datos (ubicación de los mismos, algoritmo idóneo para su captura, etc.) creando un interface abstracto (los programas que controlan la base) entran estos usuarios y los datos.

Ahora bien, la utilización de la base de datos ¿Complica el sistema o lo simplifica?. En la gráfica de la página 108 obtenemos la respuesta a esta pregunta. En dicha gráfica se representa cómo se complica el sistema en función del número de datos y el número de los diferentes usuarios de los datos, tanto con base de datos como sin ella. Como se puede apreciar, para poca cantidad de datos, el estructurarlos en una base, hace aumentar la complejidad del sistema. Ahora bien, al ir aumentando la cantidad, llega un punto en que el estructurarlos en base reduce grandemente la complejidad.

COMPARACION DE METODOS DE MANEJO DE DATOS



Antes de analizar en qué consiste la base de datos, veamos las ventajas que presenta su utilización. Consideremos varias FMM's que comparten un dato; se intenta que:

- Todas las Fmm accedan de idéntica forma al dato, independientemente de si están en el mismo o distinto procesador (con esto se pretende conseguir la independencia del diseño de las FMM's respecto a su ubicación en el sistema). La Base de Datos debe permitir por tanto un Acceso Normalizado de Datos.

  Se pretende por otra parte que cuando una FMM tiene que accesar a un dato, no conozca como está éste en memoria (si el acceso se realiza mediante punteros, tablas indexadas, etc.) ni cual es el algoritmo más idóneo a utilizar para conseguir el dato. De esta forma, si en un momento determinado se cambia toda la estructura de la Base de Datos, no es necesario cambiar las FMM's; con ello se consigue una Independencia de los Programas Respecto a los Datos.

Para conseguir este acceso normalizado a los datos y esta independencia de los programas respecto a los datos, existen los "Programas de Control de la Base de Datos" que sirven de interface entre los datos y usuarios; de tal forma que cuando una FMM del sistema quiere obtener un dato de los almacenados en la base, tiene que pedir a estos programas que se lo proporcionen.

La misión de estos programas es, por tanto, aislar la problemática de los datos (característicos de los mismos, disposición en memoria, algoritmo idóneo para obtenerlos, etc.) del resto de los programas del sistema, constituyendo con los datos una máquina virtual.

Esta idea aparece resumida en la figura de la página 111 donde se ha querido indicar cómo las FMM's deben de acceder a los datos a través únicamente de estos programas de interface.

Como se puede apreciar en la misma figura, los datos pueden estar físicamente ubicados en la memoria principal de los diferentes procesadores del sistema, y en las memorias de masas (discos y cinta).

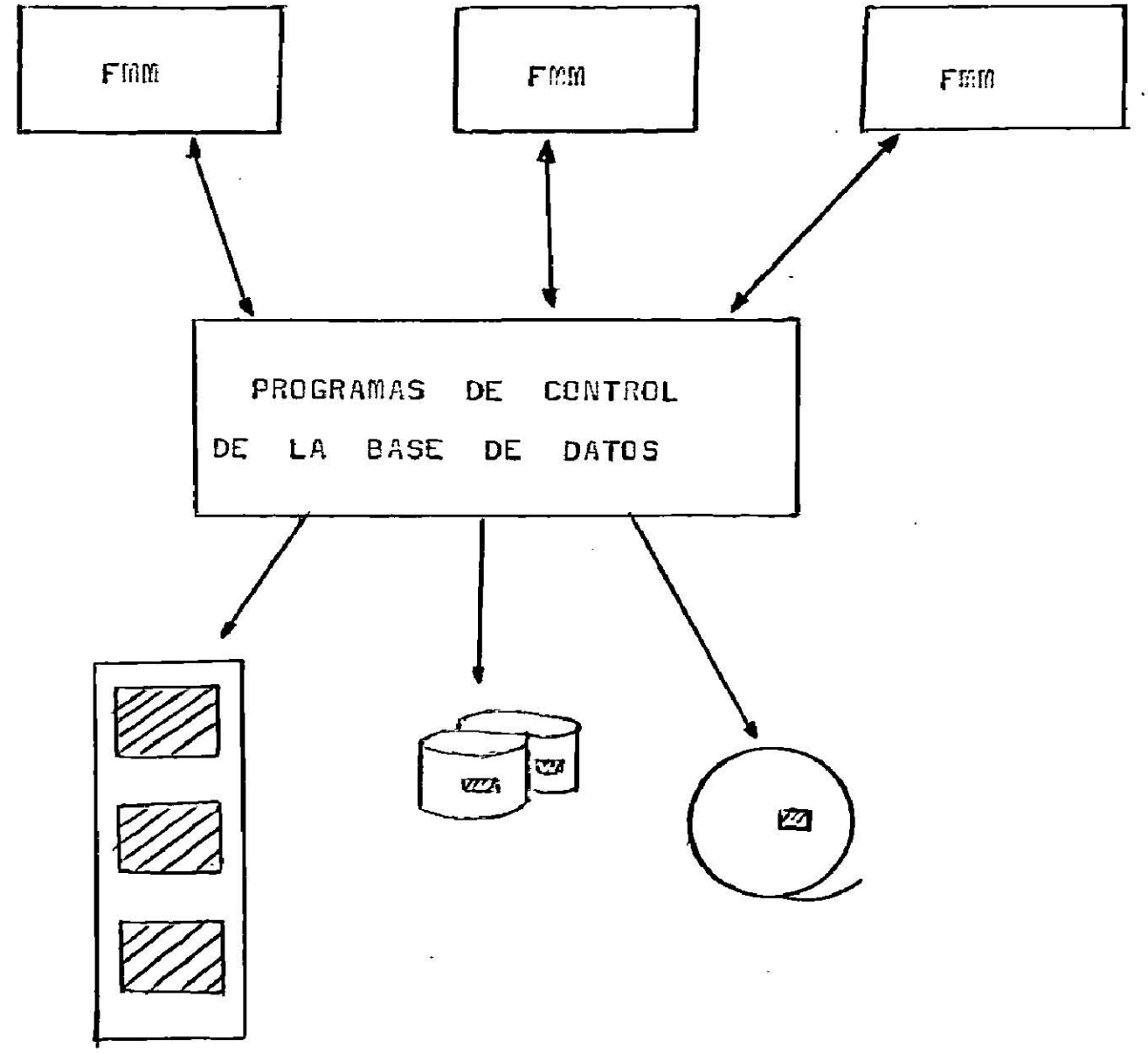
Además de los programas de control de la base de datos dentro de este subsistema encontramos otros programas con diferentes misiones, pero antes de analizar con detenimiento el subsistema, analicemos los objetivos que se han pretendido conseguir en el diseño del mismo.

Objetivos de Diseño de la Base de Datos.

A la hora de diseñar la base de datos, teniendo en cuenta las características de control distribuído del 1240, los objetivos que se persiguen son los siguientes:

- Localizar los datos de forma óptima.
  - Lo más cerca posible de los usuarios (FMM's que los utilizan) a fin de conseguir un rápido acceso.

BASE DE DATOS - VISION GENERAL



MEMORIA PRINCIPAL
DE LOS DIFERENTES
ELEMENTOS DE CONTROL

- Replicar los datos en la memoria principal de determinados CE's de tal forma los diversos usuarios de los datos accedan a ellos con la misma velocidad.
- Almacenar en disco aquellos datos que necesiten un rápido acceso.

- Proporcionar un acceso normalizado a los datos:
  - Esto se consigue con la utilización por parte de los usuarios de los datos, de lenguaje DML.
- Controlar y reorganizar las áreas de memoria disponible.
  - Existen unas áreas de memoria disponibles para almacenar los datos. Estas áreas no están asociadas a datos particulares sino que serán asignados a aquellos que dinámicamente crezcan más de prisa.
- Proporcionar seguridad a los datos.
  - Los programas que manejan la base de datos tienen que proporcionar unos mecanismos tales que ante un caso de fallo, los datos se puedan recuperar. Para ello existirán copias de los datos que necesiten ser recuperados en ficheros de seguridad, almacenados en disco. Todo esto ajeno al usuario del dato.
- Simplificar la inicialización y la extensión de la central.
  - Los datos semipermanentes se van a introducir

en la memoria de los diferentes procesadores de forma standard, simplificándose la inicialización y extensión.

Base de Datos Relacional.

La base de datos utilizada en el sistema 1240 tiene una estructura relacional, donde los datos se asocian entre sí formando relaciones.

Una RELACION es una matriz bidimensional, como se puede apreciar, la relación constituye lo que se denomina "TUPLA" y cada columna "DOMINIO". Todas las tuplas de una relación tienen la misma estructura.

En la figura de la página 114 encontramos un ejemplo de relación. En ella se ha representado la relación "Clase de Línea" utilizada en el sistema 1240. Hay que hacer notar que no se ha representado la relación real sino una simplificación pedagógica. En esta simplificación encontramos una relación con 5 dominios (procesador, terminal, señalización, categoría, directorio), compuesta por una serie de tuplas.

Se denomina clave al dominio o grupo de dominios definen unívocamente a una tupla (en el ejemplo no puede existir 2 tuplas que tengan el mismo valor de procesador y terminal ya que se ha supuesto que estos 2 dominios constituyen la clave).

EJEMPLO DE RELACION: CLASE DE LINEA

DOMINIO 1 DOMINIO 2

PROCESADOR	TERMINAL	SEÑALIZACION	CATEGORIA	DIRECTORIO
3	0	DISCO	PREVIO PAGO	7405
3	1	TECLADO	NORMAL	7601
3	2	MIXTO	CENTRALITA	7503
4	0	DISCO	NORMAL	7351

CLAVE

Diferentes Modelos de Datos.

Para los usuarios (FMM's del sistema que van a utilizar la base de datos) los datos son relaciones de N tuplas a las cuales pueden acceder de determinada forma y modoficar. A esta visión de los usuarios es a lo que se llama MODELO LOGICO.

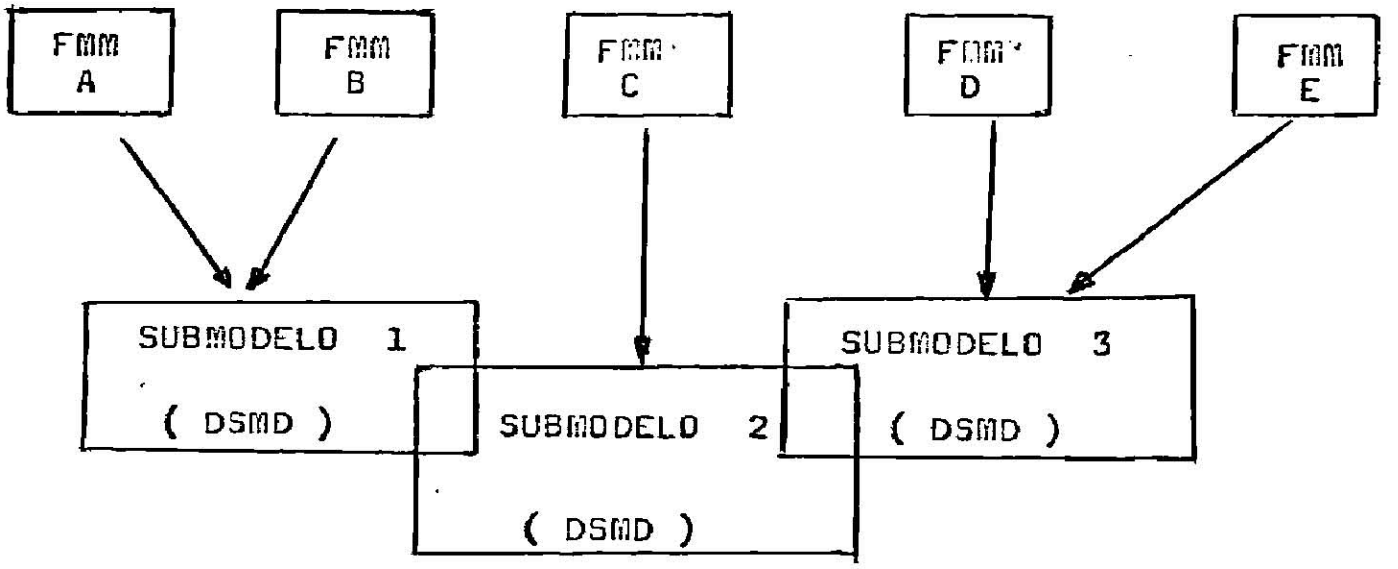
Ahora bien, estas relaciones se encuentran almacenadas en determinadas posiciones de memoria (tanto en la memoria principal de los micros como en la de masas), con una estructura determinada. A la organizaciónfísica utilizada para almacenar los datos, se le llama MODELO FISICO.

Gracias a la utilización del Modelo Lógico, las FMM ven los datos de una forma mucho más abstracta que como esos datos están realmente almacenados en memoria, Modelo Físico.

Por otra parte, una FMM no necesita acceder a todas las relaciones que forman la base de datos sino solamente a determinadas de ellas (que dependen de la función que realiza la FMM). Al conjunto de relaciones utilizadas por un usuario se le denomina SUBMODELO. En este sentido, el modelo lógico será la unión de todos los submodelos.

En la figura de la página 116 aparecen reflejados los diferentes modelos de Datos.

MODELOS DE DATOS



MODELO LOGICO
(DMD)

DISPOSICION DE LOS DATOS EN MEMORIA

MODELO FISICO
(PDM)

Ubicación de los Datos.

Nos enfrentamos ahora con el problema de donde colocar los datos en memoria; la primera idea es ponerlos lo más cerca posible de la FMM que los vaya a utilizar, siguiendo en la idea ya expuesta como objetivo de diseño.

Por otra parte ¿se ponen los datos que constituyen la base de datos sólo en la memoria de los diferentes procesadores del sistema? ¿en memoria y el disco para que en caso de caída del procesador, no se pierdan los datos? ¿solo en disco?.

Para resolver todo ésto están los siguientes criterios:

a. Los datos que se accede a ellos con frecuencia, tanto para leerlos como para modificarlos y que en caso de caída de un procesador no hace falta que recuperen sus valores, se colocan exclusivamente en la memoria principal de los diferentes procesadores del sistema.

Ejemplos de estos datos son:

- Estados de las líneas: libre, ocupada en tratamiento de llamadas, en mantenimiento, etc. Se encuentran en el TCE de las correspondientes líneas (siguiendo la idea indicada anteriormente de ponerlos lo más cerca posible de donde se vayan a utilizar). Observar que en caso de caída del micro, cuando el micro queda de nuevo operacional, las líneas se ponen como en libres no haciendo falta recobrar el valor que tenían cuando cayó el CE.

- Estado de los emisores y receptores: ACE el sistema En este caso, como los ACE's de sistema se encuentran siempre en parejas la estrategia de activo/pasivo, cuando se

hable de que un dato se encuentra en el paso del sistema, se sobreentiende que se encuentra en los 2 ACE's que forman la pareja; de forma que, cuando se modifica en uno de ellos es necesario modificarlo también en el otro. Este tipo de redundancia como en el tipo que veremos más adelante son los "programas de la base de datos" los que se encargaban de asegurar la "coherencia" (el mismo dato, almacenado en varios lugares, debe tener el mismo valor en todos ellos).

b. Los datos que se leen con frecuencia pero que rara vez se modifican (semipermanentes) y que en caso de caída del procesador donde están almacenados hace falta recuperar sus anteriores valores, se colocan en la memoria principal del procesador adecuado, más en disco para poderlo recuperar.

Ejemplos de estos datos son:

- Clase de línea: (simplificado en la relación de la figura de la página 114). Se encuentra almacenado en el TCE de las correspondientes líneas y se cambiará a petición del operador. En caso de caída del procesador será necesario cargar de nuevo estos datos en la memoria principal.

- Tablas de encaminamiento: ACE's del sistema con la salvedad indicada anteriormente sobre duplicidad.

Estos datos pertenecen al conjunto de "datos semipermanentes" (SPD) de la central, cuyo valor se va a modificar muy raramente.

De nuevo, los "programas de la base de datos" se encargan de que todas las copias del mismo dato tengan el mismo valor.

c. Por último los datos que son muy voluminosos o se utilizan muy raramente se almacenan solamente en disco.

Ejemplos de estos datos son:

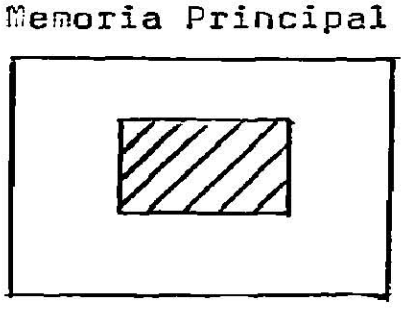
- Tabla de errores de mantenimiento.
- Formatos por diálogos hombre-máquina.

En las figuras de las páginas 120, 121 y 122 aparecen resumidas estas ideas.

En la figura de la página 123 aparecen diferentes datos asociados a los diferentes procesadores del sistema.

UBICACION DE LOS DATOS

1) DATOS ALMACENADOS SOLAMENTE EN LA MEMORIA PRINCIPAL DE LOS DIFERENTES PROCESADORES.



CARACTERISTICAS:

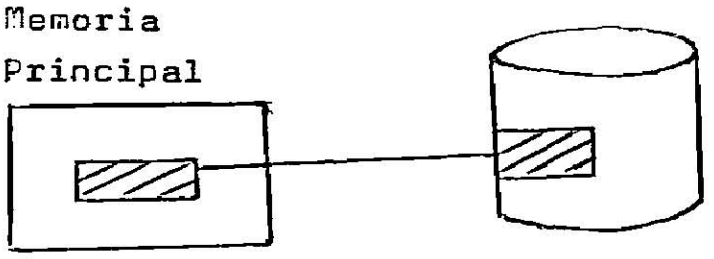
- Datos a los que se accede con frecuencia en lectura y escritura.

EJEMPLOS:

- Estado de los enlaces de salida.
- Estado de los emisores y receptores.
- Estado de las líneas.

UBICACION DE LOS DATOS

2) DATOS ALMACENADOS EN MEMORIA PRINCIPAL DE LOS PROCESADORES CON COPIA EN MEMORIA DE MASAS.



CARACTERISTICA:

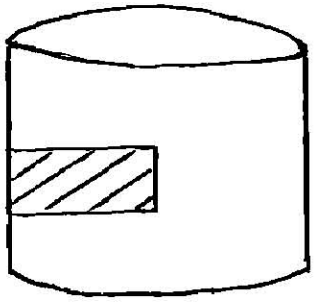
- Datos que se lee con frecuencia, se modifican poco y necesitan recuperación en caso de error.

EJEMPLOS:

- Clase de servicio.
- Enrutamiento.
- Clase de línea.

UBICACION DE LOS DATOS

3) DATOS ALMACENADOS SOLAMENTE EN DISCO



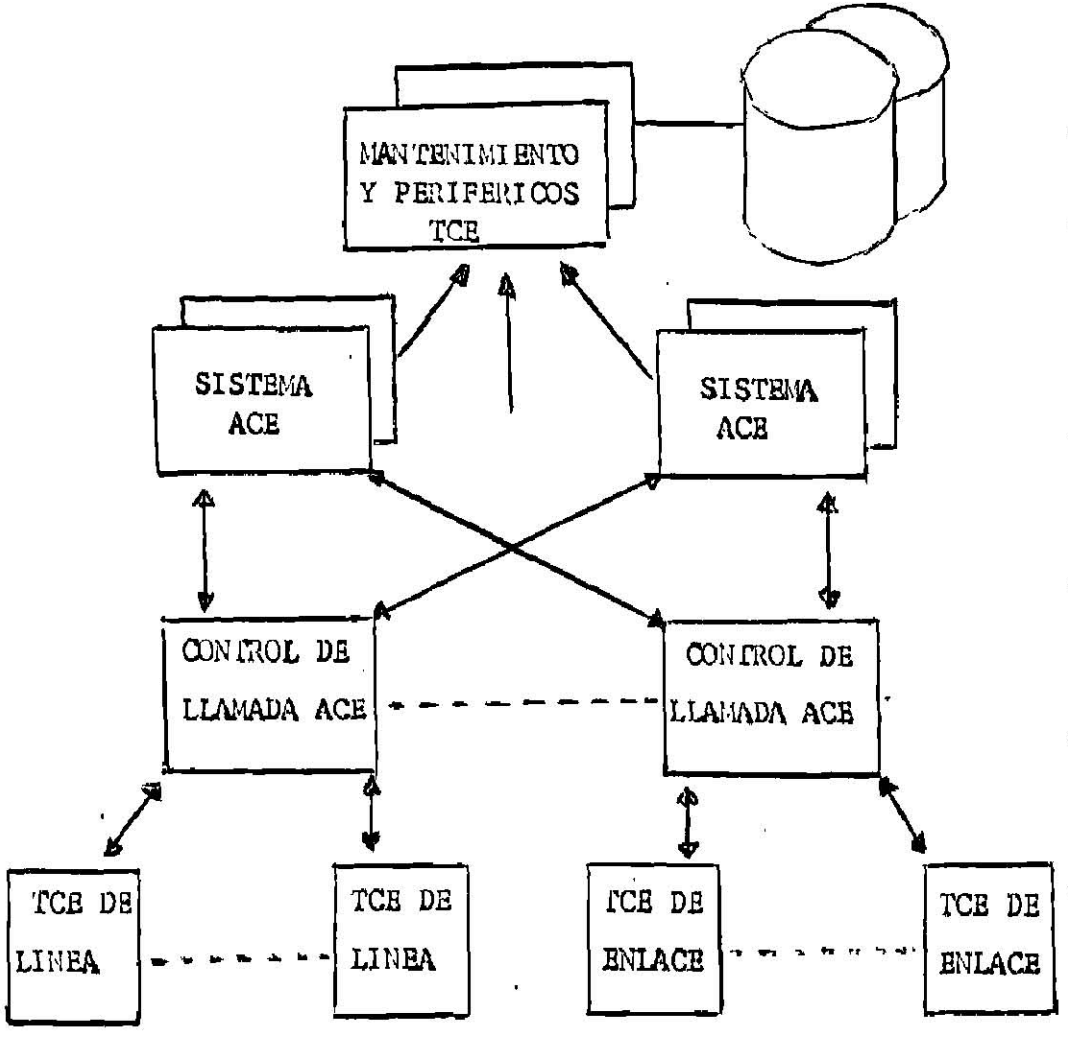
CARACTERISTICAS:

- Datos que se usan poco y son voluminosos.

EJEMPLOS:

- Errores de mantenimiento.
- Formatos para diálogos hombre-máquina.

UBICACION DE LOS DATOS



- DATOS DE MANTENIMIENTO COMUNES

- DISCO - COPIA DE DATOS CRITICOS
  - DATOS USADOS RARAMENTE
  - DATOS MUY VOLUMINOSOS

- DATOS COMUNES DE CONTROL DE LLAMADAS
(ENCAMINAMIENTO, TAREAS)

- DATOS DINAMICOS CON ALTA SEGURIDAD
(TARIFICACIONES, ESTADO ENLACES SALIDA

- DATOS RELATIVOS A LOS TERMINALES PARA CONTROL DE LLAMADAS, ADMINISTRACION

- DATOS RELATIVOS A LOS TERMINALES PARA SEÑALIZACIÓN Y MANEJO DE DISPOSITIVOS
(CARACTERISTICAS DE LOS DISPOSITIVOS)

Diferentes Elementos de la Base de Datos.

Los diferentes elementos que constituyen la base de datos, considerando tanto los datos en sí como los programas, son los siguientes:

a. Datos: contenido de los mismos e información necesaria para su acceso (directorios y diccionarios).

b. Sistema de Control de la Base de Datos. (DBCS): encargado de proporcionar el acceso a los datos, de los diferentes usuarios (la petición de acceso se hace mediante el comando del lenguaje DML, encargándose el DBCS de la ejecución de dichos comandos).

c. Sistema de seguridad de la Base de Datos. (DBSS): encargado de proporcionar los mecanismos de seguridad necesarios para no perder datos.

d. Sistema de Organización de la Base de Datos. (DBOS): permite la inicialización de los datos y soporta las extensiones de la central.

e. Administración de la Base de Datos. (DBA): obtiene de los diseñadores la información de los datos que éstos necesiten y defina los diferentes modelos. (utilizando el lenguaje DDL).

Se han citado dos lenguajes (DML y DDL) utilizado en este subsistema de forma muy diferente.

DDL Data Definition Language.
Utilizado para definir los diferentes modelos de datos.

DML Data Manipulation Language. Utilizado por las FMM's/SSM's para indicar al DBCS la operación que quieren realizar sobre los datos (obtener un dato, modificarlo, etc.)

Sistema de Control de la Base de Datos (DBCS).

Conjunto de programas (FMM's y SSM's) que soportan, en tiempo de ejecución, las operaciones sobre la base de datos solicitados por las FMM's y SSM's del sistema.

Al igual que el S.O. se encuentra almacenado en todos los CE's, siendo el de los L/T TCE's un subconjunto del general.

Las FMM's y SSM's realizan sus peticiones mediante sentencias del lenguaje DML. Estas sentencias que serán procesadas y trasladadas a CHILL antes de ser compilados los diferentes programas, serán ejecutadas por el DBCS. Realmente las sentencias DML constituyen el interface entre los programas de aplicación y la Base de Datos.

Sentencias DML.

Las diferentes operaciones que las FMM's pueden realizar sobre los datos son las siguientes:

- Leer una tupla.
- Modificar sus valores.
- Borrar una tupla.
- Crear una nueva tupla.
- Retener tuplas o relaciones durante un determinado tiempo, impidiendo el acceso de los restantes programas a esta tupla o relación.
- Liberar lo anteriormente retenido.
- Proporcionar una serie de funciones elementales.

Desde el punto de vista de una FMM, la forma de trabajo es la siguiente:

Una FMM tiene acceso permitido a un submodelo pero para poder utilizarlo, antes de emitir ninguna sentencia de ejecución, debe definirlo. Esto se consigue gracias a las sentencias DML "INVOKE"

INVOKE Submodelo - x DSMD

Al preprocesar esta sentencia se crean las sentencias CHILL apropiadas para definir los datos del submodelo (sentencias DML CHILL)

Estas sentencias crean una zona donde se pueden cargar las tuplas de las diferentes relaciones que pertenecen al submodelo invocado por el usuario. (una tupla por relación). A esta zona se le denomina User Working Area.

Una vez creada la UWA ya se pueden pedir al DBCS las restantes funciones sobre los datos del submodelo.

Ejemplo: GET CLASE - DE - LINEA WHERE

(PROCESADOR= 3, TERMINAL= 0) esta sentencia (si se utiliza la relación:

CLASE DE LINEA representada en la figura de la página 114) cargaría en la parte de la UWA reservada para cargar la tupla de esta relación, los valores:

3	0	DISCO	PREVIO PAGO	7405

Observar que para definir la tupla se ha dado una condición que cumple la misma. Esta condición puede satisfacerla más de una tupla de la relación, obteniendo en este caso la primera de ellas (solo se puede cargar una en la UWA).

Modificar Dato (MODIFY): Modifica los valores de los domi-

nios de las distintas tuplas excepto de los que son o forman parte de la clave (dado que la clave define unívocamente la tupla, cambiar sus valores implica definir una tupla diferente, cosa no permitida con esta sentencia).

Ejemplo: MODIFY - DE - LINEA WHERE
(PROCESADOR = 3, TERMINAL = 0 TO DIRECTORIO = 7820); el efecto de esta sentencia es, como se ve, modificar el valor del dominio "directorio" de la primera tupla de la figura de la página 114.

MODIFY CLASE - DE - LINEA WHERE
(PROCESADOR = 4, TERMINAL = 0) TO
(TERMINAL = 6); esta sentencia no es válida ya que al intentar cambiar la clave, lo que realmente está haciendo es cambiar la tupla, cosa no admitida en este tipo de sentencia.

Borrar Tupla (DELETE): Borra la tupla que cumple la condición establecida. Mientras que en las dos sentencias anteriores la condición no tenía por qué cumplirla una única tupla (obteniendo el DBCS la primera que encontrara que satisfaciera la condición) al utilizar la sentencia de borrar, la condición establecida debe cumplirla solamente una tupla (o ninguna). Al encontrar la tupla el DBCS la borra de la relación.

Ejemplo: DELETE CLASE - DE - LINEA WHERE.
(PROCESADOR = 3); no válida porque hay más de una tupla de esa relación que cumplen esa condición.

DELETE CLASE - DE - LINEA WHERE (PROCESADOR = 3, TERMINAL = 0, SEÑALIZACION TECLADO); sentencia válida pero sin efecto ya que en la relación (figura de la página 114) no existe ninguna tupla que cumpla todas esas condiciones.

Crear Tupla (STORE): Previamente ha debido crear en la UWA la tupla deseada (los valores dados al dominio o dominios) de la clave deben ser distintos a todas las tuplas ya existentes) y al decir: STORE nombre - de - la relación el DBCS toma de la UWA la tupla y la almacena en la relación correspondiente.

Retener y Liberar Tuplas (HOLD RELEASE): Si un programa necesita asegurarse que mientras él está trabajando con determinada tupla o relación ningún otro programa entre a leerla o modificarla, puede pedir al DBCS que la retenga impidiendo el acceso a la misma.

Si se trata de retener tupla, al pedir un GET a la tupla puede indicar que se retenga esa tupla.

HOLD: Si lo que pretende es retener una relación entera el programa escribirá:

HOLD nombre de la relación y, cuando ya no la necesite:

· Funciones.

Finalmente, las funciones que una FMM puede pedir y el DBCS proporcionar ; son:

Contar (COUNT): Determina el número de tuplas que cumplen una determinada condición.

Ejemplo: NUM: = COUNT (CLASE - DE - LINEA WHERE)
(SEÑALIZACION = DISCO)

NUM: = COUNT (CLASE - DE - LINEA)

En el segundo ejemplo, la variable NUM. tiene el número de tuplas que contiene la relación:
CLASE - DE - LINEA.

Sumar (SUM): Los valores de un determinado dominio de las tuplas de una relación que cumple determinada condición.

Calcular el valor máximo (MAX)

Calcular el valor mínimo (MIN)

Calcular el valor medio (AUG)

Como se puede apreciar, el acceso de las FMM's a los datos es muy sencillo utilizando el DML.

Estas sentencias DML son unas llamadas a los programas de control de la base de datos para que éstos se encarguen de realizar esas funciones.

Estructura del DBCS.

Antes de analizar la estructura interna que posee el DBCS para poder realizar su función, veamos los diferentes tipos de relaciones que pueden existir.

RELACION LOCAL: El usuario (FMM que utiliza la relación), se encuentra almacenado en el mismo CE que la relación.

RELACION GLOBAL: El usuario se encuentra almacenado en distinto CE que la relación.

RELACION DISTRIBUIDA: Las diferentes tuplas de la relación se encuentran almacenadas en más de un elemento de control por ejemplo: la relación Clase - de - Línea se encuentra distribuída en todos los TCE's (en cada uno de ellos las tuplas correspondientes a los terminales asociados a ese TCE).

RELACION REPLICADA: La misma relación almacenada en más de un elemento de control por ejemplo: las relaciones de los ACE's de los sistemas estarán almacenadas en los ACE's de la pareja.

REALACION REAL: La visión lógica de todos los dominios de una tupla coincide con la relación física.

RELACION VIRTUAL: La visión lógica es una combinación de relaciones reales. (por ejemplo si la relación física contiene más información que la que necesita un usuario, puede ser inte-

resante obtener una relación virtual).

Veamos a continuación cómo está constituída el DBCS para poder realizar su función.

Cuando una FMM quiere obtener un dato, le pide al CE que se lo proporcione. Esta petición la realiza mediante un comando DML. No olvidar que la FMM no sabe donde está el dato, es decir no sabe si este dato es local o global, por tanto en el comando DML no va esta información.

El DBCS recoge la petición, analiza si el dato es local o global, lo busca (en su CE o en el CE correspondiente) y se lo proporciona a la FMM que lo solicite.

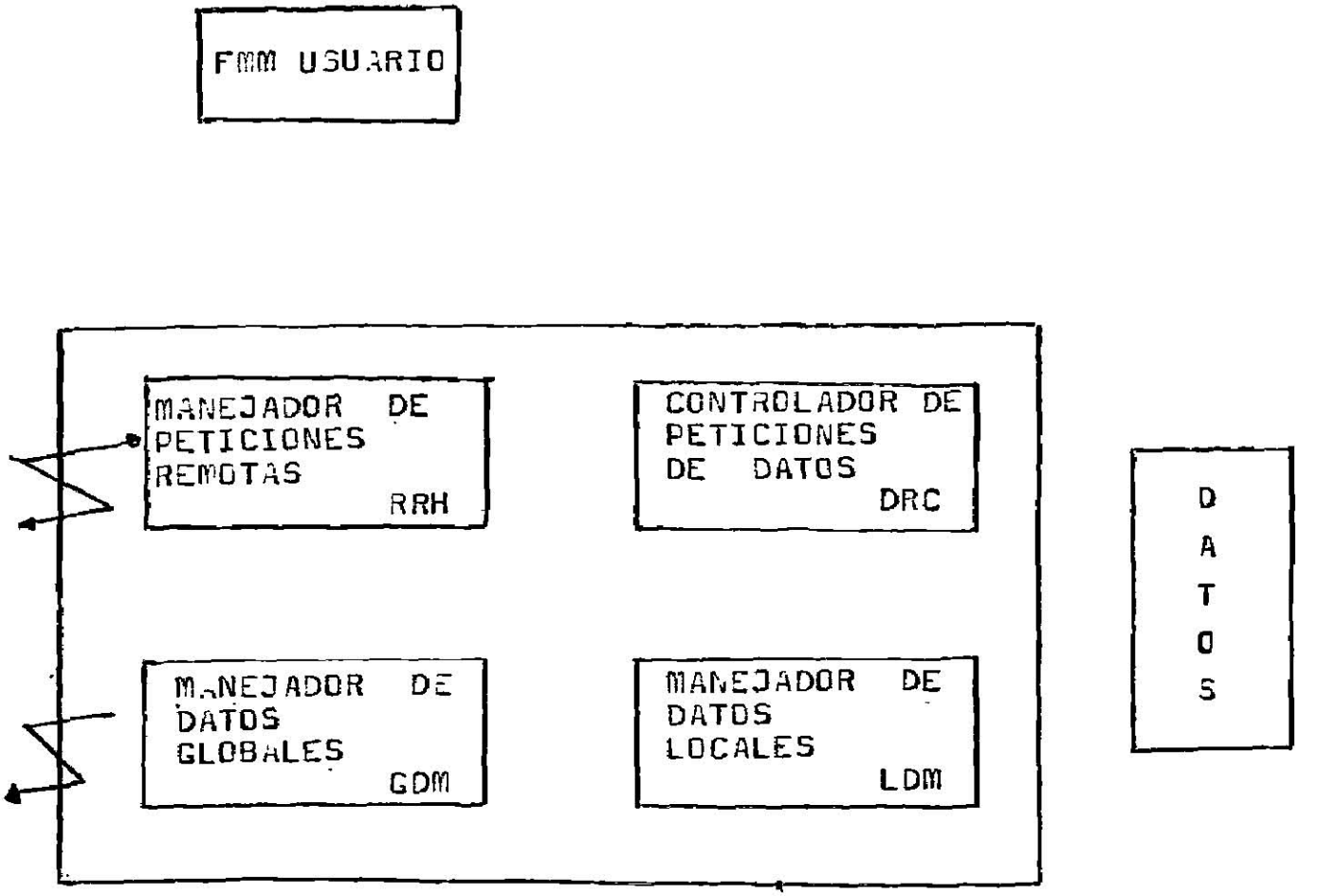
Para poder realizar esto, el sistema de control DBCS se encuentra descompuesto en varios módulos como aparece en la figura de la página 133.

La función de analizar la petición la realiza el "Controlador de peticiones de datos" (Data Request Controller). Para ello mira en unas tablas (directorios) que le indican si el dato esta en ese CE (dato local) o en otro CE (dato global).

- Si es local, le pasa la petición al "Manejador de datos Locales" (Local Data Manager) el cual toma el dato y se lo devuelve al controlador el cual se lo pasa a la FMM que lo solicitó.

- Si es global, le pasa la petición al "Manejador de datos Globales" (Global Data Manager) el cual se encargará de ir a otro CE a buscar el dato a tra-

ESTRUCTURA DEL SISTEMA DE CONTROL DE LA BASE DE DATOS



DRC = Data Request Controller

LDM = Local Data Manager

GDM = Global Data Manager

RRH = Remote Request Handler

vés de un mensaje. En otro CE el mensaje lo recibirá una FMM perteneciente al DBCS denominada "Manejador de Peticiones Remotas" que se lo pasará al "Controlador de Peticiones de Datos" que volverá de nuevo a analizar si el dato es global o local.

En la figura de la página 135 se ha dibujado un escenario en donde se ve los pasos que se dan para conseguir un dato local y en la figura de la página 136 el acceso a un dato global.

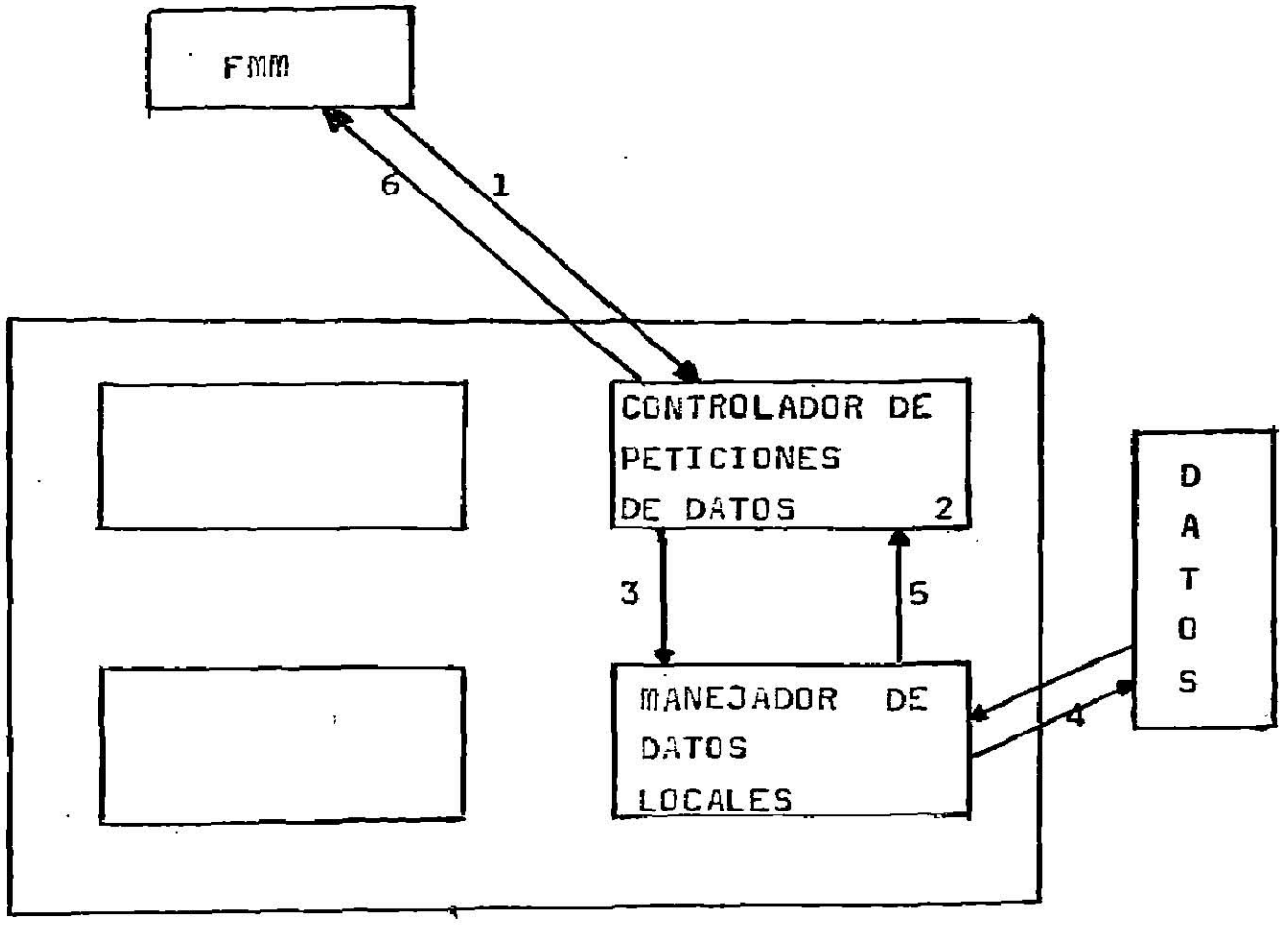
En los L/T TCE's en donde como se ha indicado anteriormente, existe un DBCS menos potente que el de los otros CE's, no está permitido el acceso a los datos globales, así la parte "MANEJADOR DE DATOS GLOBALES" de la figura de la página 133 no existirá en este DBCS. Por otra parte, sí admite peticiones de datos locales desde otros CE's, por lo que la FMM "Manejador de peticiones remotas" sí que debe existir.

El DBCS de los procesadores duplicados (ACE's sistemas y P & M posee un módulo más como lo muestra la figura de la página 138). Ello es debido a que cuando se quiere escribir en relaciones locales a sus CE's, es necesario hacerlo en ambos CE's que componen la pareja y de asegurar esta duplicidad se encarga la FMM "Controlador de peticiones replicadas".

Analicemos ahora en qué consisten los directorios y diccionarios necesitados por el DBCS para poder encaminar adecuadamente las diferentes peticiones.

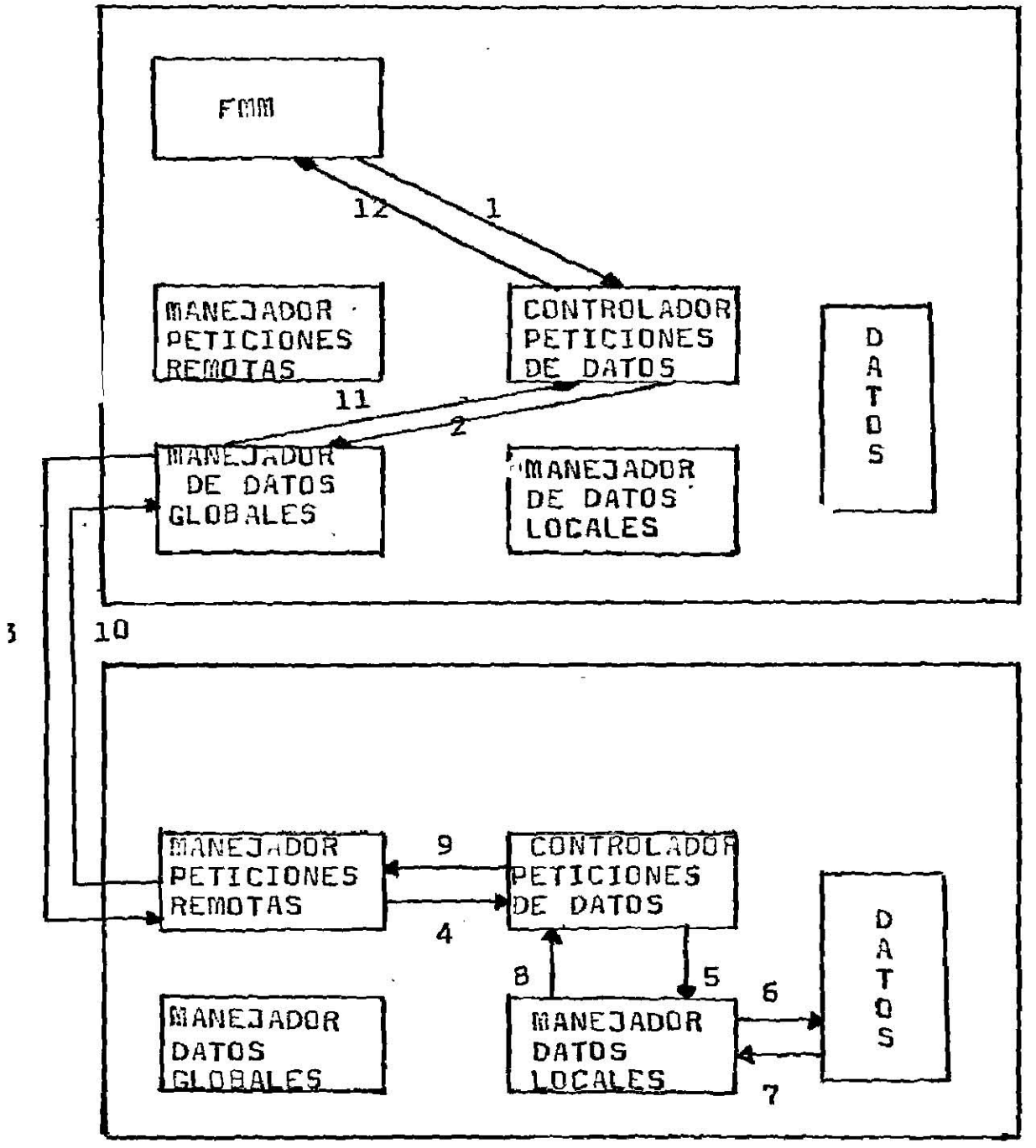
El DBCS posee para cada relación a las que pueden acceder las diferentes FMM's y SSM's, una indicación de si se trata de relación global (almacenada en otro CE) o local. A esta información se

ACCESO A UN DATO LOCAL



1. Petición de una Fmm al DBCS de un dato.
2. Búsqueda en directorios y diccionarios para analizar si el dato pedido es local o global.
3. Petición del dato local.
4. Acceso al dato.
5. Entrega del dato.
6. El dato se lo pasa a la FMM que lo pidió.

ACCESO A UN DATO GLOBAL



le llama directorio global como lo muestra la figura de la página 139.

En caso de tratarse de relación global, el directorio indica la identidad de CE donde se encuentra almacenada la relación. Con esta información ya puede el DBCS enviar un mensaje a ese CE para pedir el acceso al dato.

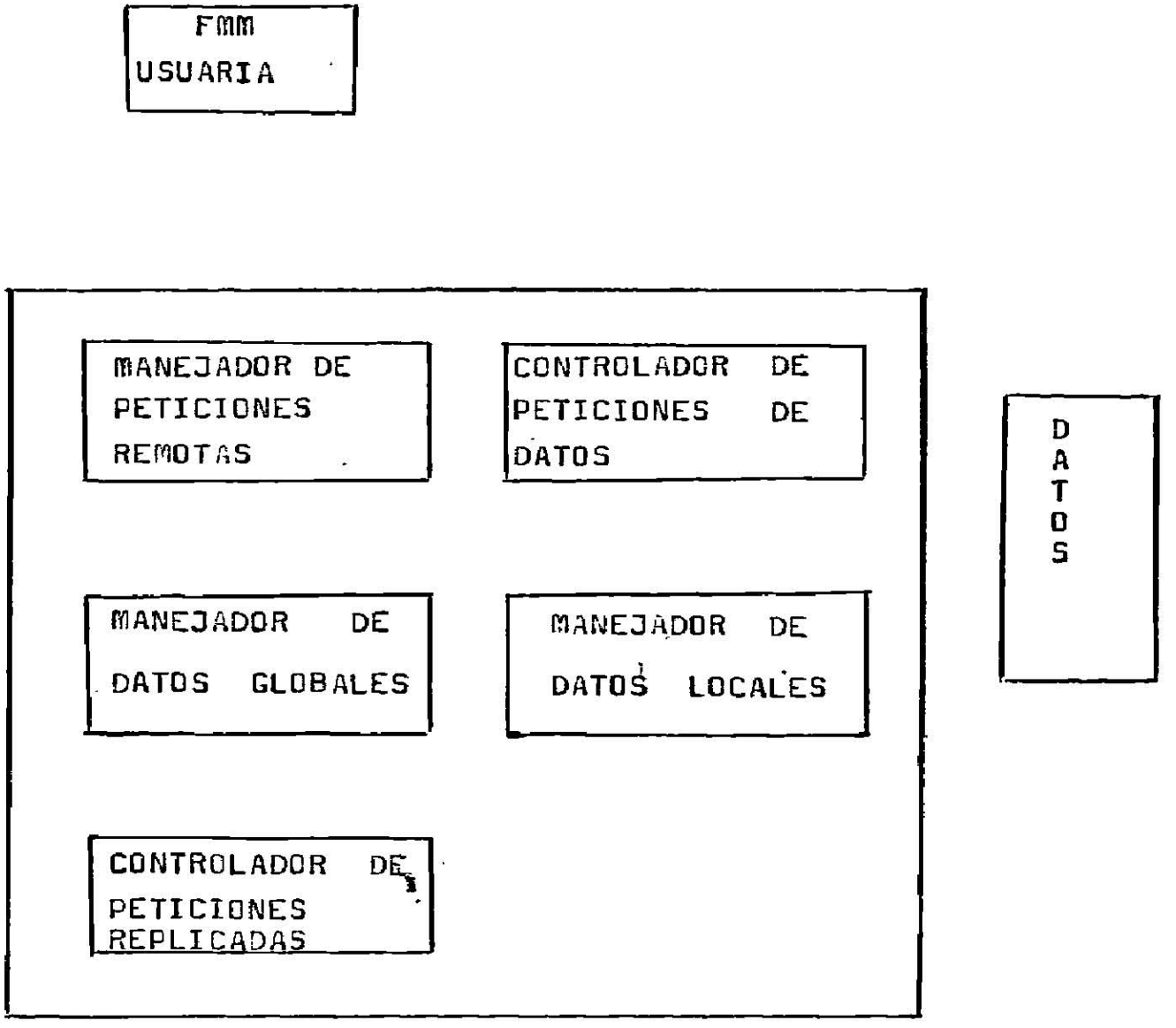
En caso de tratarse de relación local, el directorio global da el índice al directorio local.

En este directorio local se encuentra:

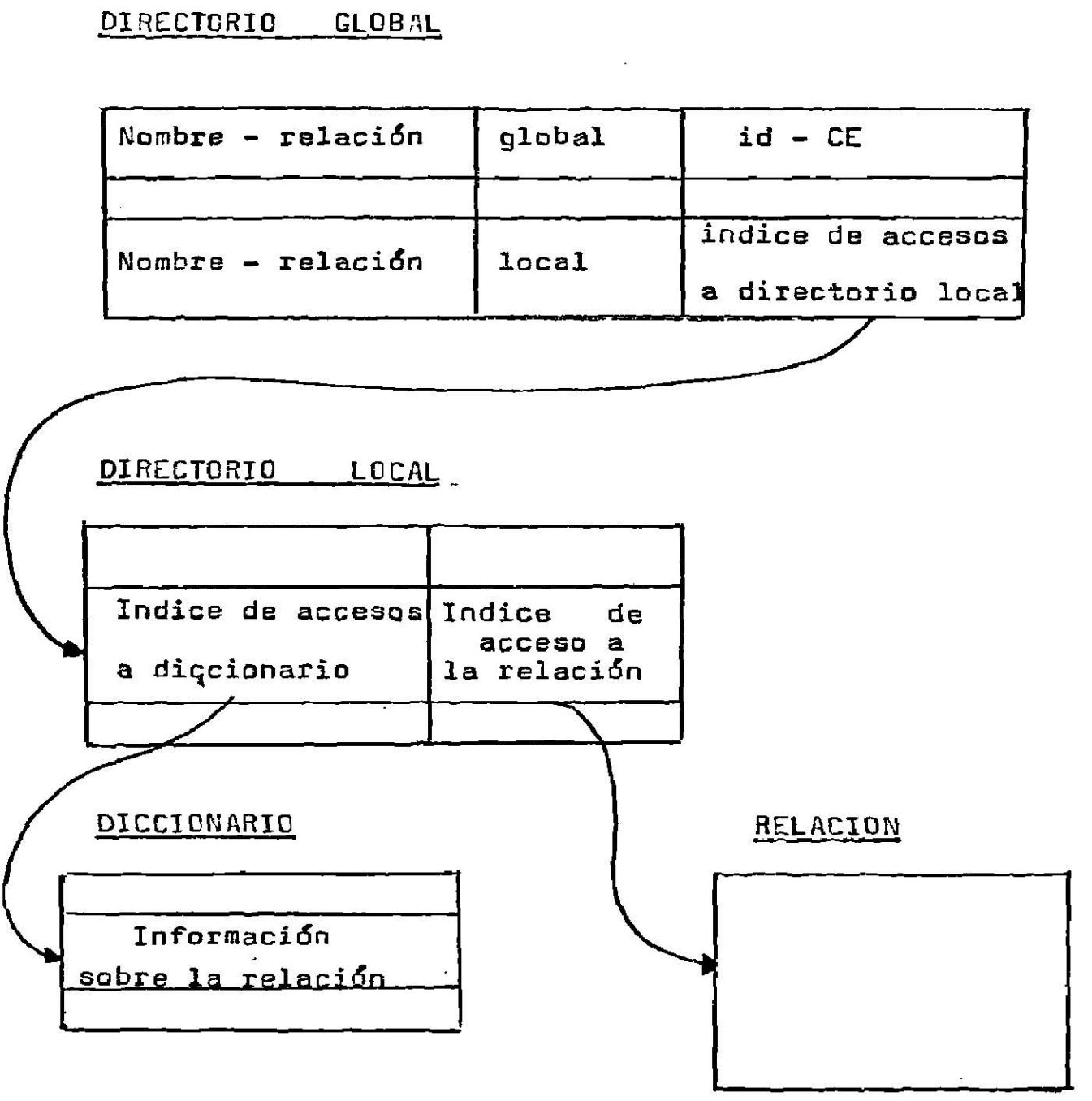
- Indice al diccionario (que contiene información sobre la relación: número de dominios y características de ellos, etc.).
- Indice para acceder a la relación.

En la figura de la página 140 encontramos un ejemplo de directorios y diccionarios para la relación CLASE - DE - SERVICIO (COS).

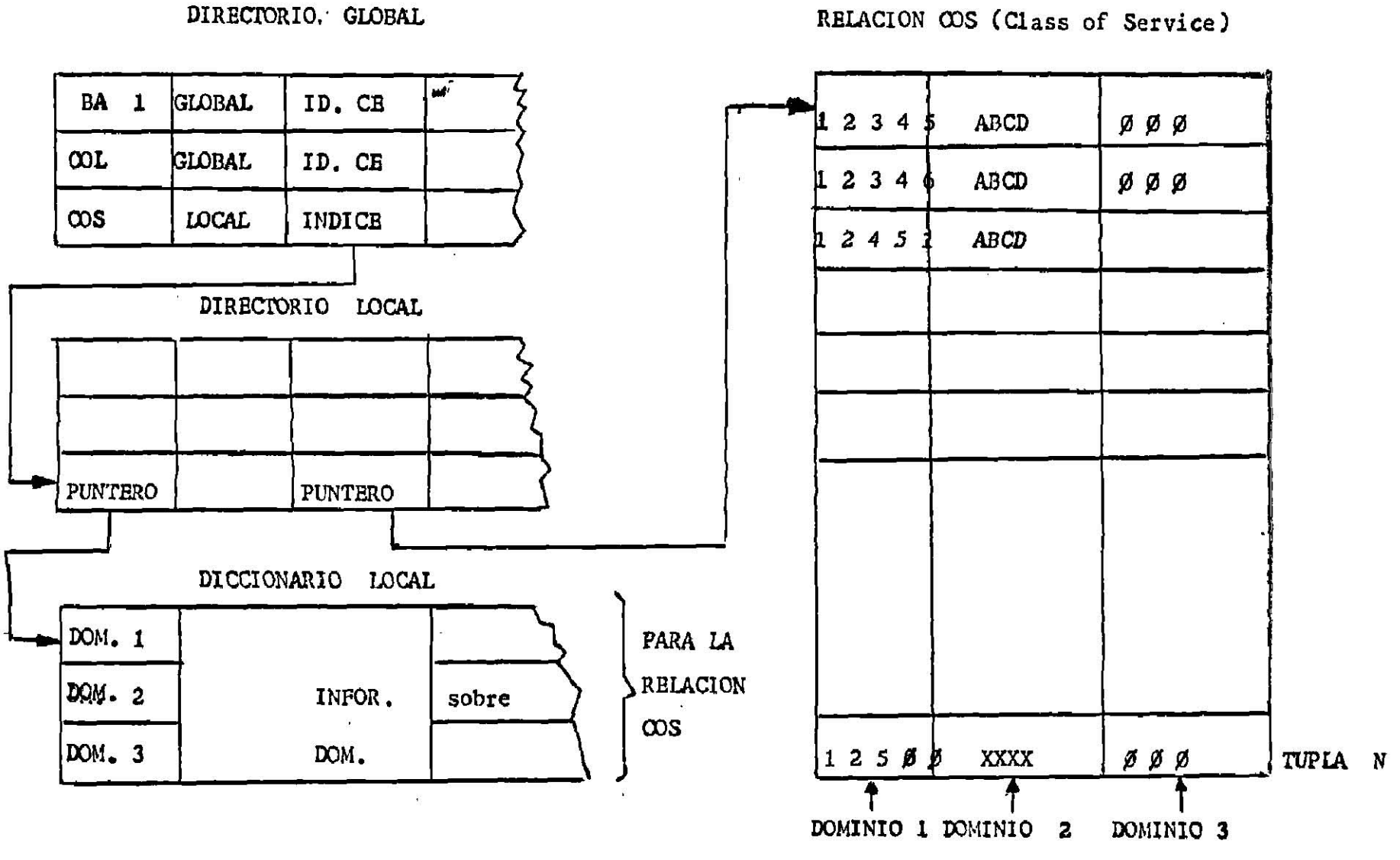
ESTRUCTURA DEL DBCS EN LOS PROCESADORES DUPLICADOS



DIRECTORIOS Y DICCIONARIOS



EJEMPLO DE DIRECTORIOS Y DICCIONARIOS



SISTEMAS DE SEGURIDAD DE LA BASE DE DATOS (DBCS)

Debido a que los datos van a estar distribuídos en las memorias de los diferentes procesadores y en las de masas es necesaria la existencia de determinados programas capaces de proporcionar mecanismos de seguridad para garantizar:

- No perder datos.
- Los datos almacenados en diferentes representaciones tienen el mismo valor.

De proporcionar estos mecanismos de seguridad se encarga el DBSS (Sistema de seguridad de la Base de Datos).

Este módulo va a estar almacenado en P & M, T C E y sus funciones van a ser las siguientes:

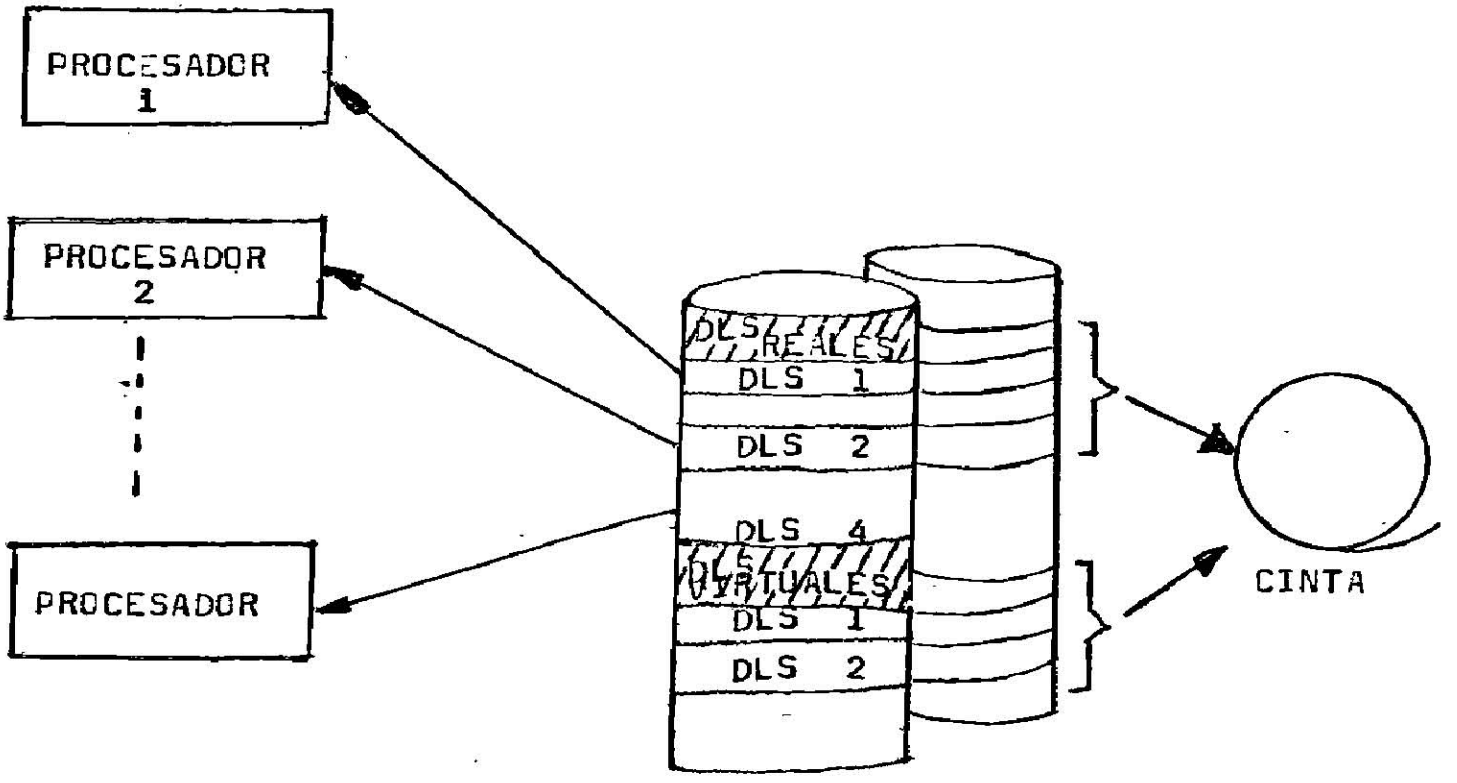
- Mantener una copia en disco de la base de datos almacenadas en los diferentes procesadores, que será actualizada cuando se modifiquen los valores de los datos de las memorias de los CE's.

- Asegurar la consistencia de los datos, manteniendo los ficheros de seguridad.

Veamos el funcionamiento del DBCS para poder realizar estas funciones.

Cada procesador del sistema está cargado con unos programas (GLS) y unos datos (DLS). En el disco debe haber una copia de todos los GLS's y DLS's de los diferentes procesadores. En la figura de la página 142 aparecen representadas las copias de los diferentes procesadores (nos limitamos a los DLS's por ser el objetivo de

ALMACENAMIENTO DE DL'S



este tema) de los DLS's, los cuales se distinguen entre DLS's reales (los que se cargan en los diferentes procesadores) y los DLS's virtuales (de aquellos datos en que al estar solamente en disco no pertenecen a ningún DLS real).

El DLS de un procesador contiene todos los datos del mismo. Parte de estos datos estarán en la base de datos (no todos ellos ya que habrá datos internos a las FMM's que no será necesario meter en la base de datos). Dentro de los datos controlados por la base de datos, parte de ellos necesitarán tener su copia en disco actualizada, de tal forma que cuando se modifica su valor en la memoria principal, hace falta modificar también su valor en el disco (A este tipo de datos se les denomina datos disk-backed).

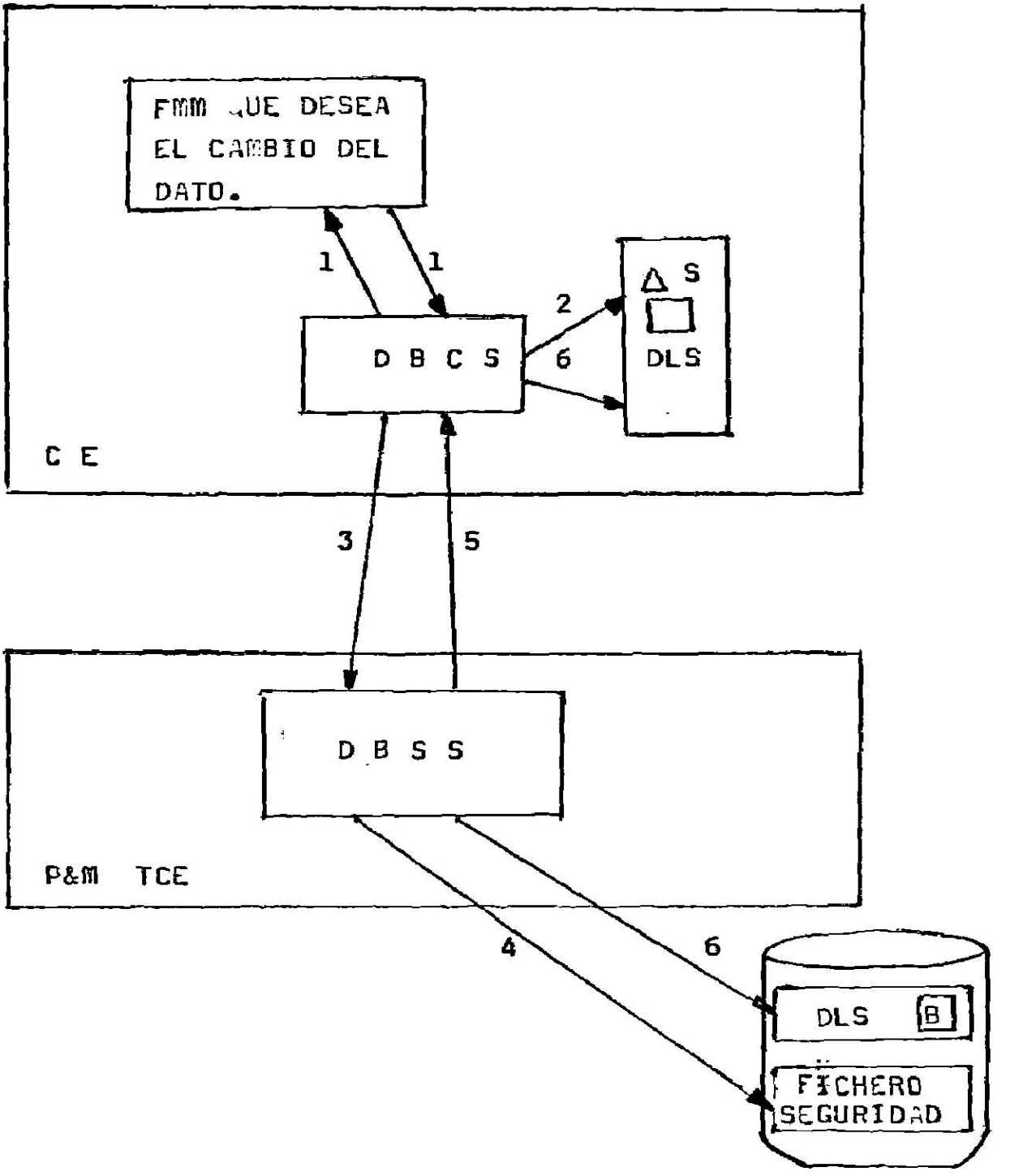
Cambio de Datos con Copia en Disco.

Las modificaciones en disco de los datos disk-backed las realiza el DBSS. En la figura de la página 144 se ha representado cómo se modifica uno de ellos, analicémosla a fin de ver la relación DBSS y DBCS.

Consideremos que en un determinado momento una FMM desea cambiar el valor del dato B de un valor B1 a un valor B2, donde el dato B es del tipo disk-backed. Supongamos para simplificar que se encuentra almacenado en el mismo CE donde está la FMM que pide su cambio.

Como este dato pertenece a la base de datos, el que debe manejarlos es el "Sistema de Control de la Base de Datos" (DBCS). Ahora bien, como queremos tener la copia en disco actualizada, cuan

ESCENARIO DE CAMBIO DE DATO CON COPIA EN DISCO



DBCS: Sistema de control de la base de datos.

DBSS: Sistema de seguridad de la base de datos.

do se procede al cambio en la memoria principal se debe proceder también al cambio en el disco y de esto se va a encargar el "Sistema de Seguridad de la Base de Datos" (DBSS) almacenado en P&M TCE. La forma de trabajo es:

1. La FMM pide, a través de la sentencia MODIFY (DML), al DBCS que cambie el valor de B de B1 a B2. Esta FMM no sabe que el dato es local ni que es disk-backed (realmente ha sido el diseñador de la FMM el que le ha indicado a los diseñadores de la base de datos que el dato B debe ser disk-backed, ya que es este diseñador el que sabe la importancia que tiene este dato. Ahora bien, no tiene que tener en cuenta ésto a la hora de pedir el cambio durante la ejecución.
   Más adelante se analizará cómo cada diseñador indica si los datos a manejar por las FMM's por el diseñador debe de ser de un tipo o de otro).
   El DBCS recibe la petición y detecta que se trata de un dato local del tipo disk-backed y que por tanto debe contactar con el DBSS para que proceda a cambiarlo en el disco.
2. El DBCS prepara el cambio sin ejecutarlo de momento.
3. El DBCS envía entonces un mensaje al DBSS indicando que es necesario modificar el dato B de B1 a B2 (observar que este mensaje lo envía antes de modificar B en la memoria principal).

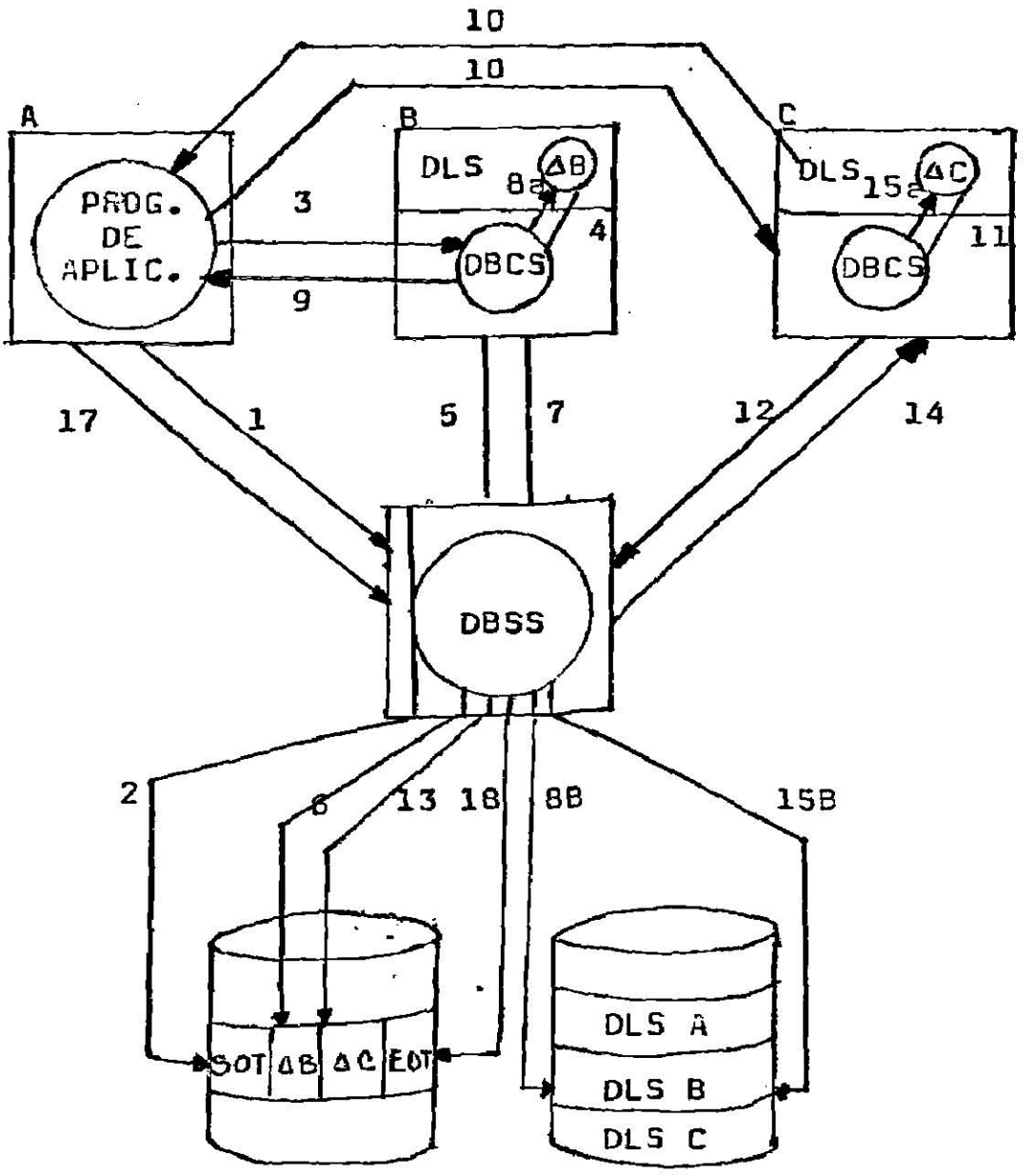
4. Al recibir la petición, el DBSS abre un fichero de seguridad en el disco anotando el valor antiguo y nuevo, a fin de que si por algún problrma no se puede cambiar alguna representación del dato B se puede cambiar alguna representación del dato B se puede devolver a éste su valor antiguo.
5. Una vez abierto el fichero de seguridad el DBSS indica al DBCS que puede proceder a cambiar.
6. El DBCS modifica el valor del dato en la memoria del procesador y el DBSS lo modifica en la copia del disco.
7. Por último se le devuelve el control a la FMM que envió la sentencia DML pidiendo el cambio.

Existen situaciones en las que el cambio de un dato no es tan simple como el expresado en el análisis de la figura de la página 144 . Cuando el dato a cambiar implica una relación distribuída en más de un procesador, la situación se complica. a esto se le llama TRANSACCIONES DE ALTO NIVEL, y como ejemplo podemos considerar cuando se desea modificar la clase de línea que se encuentra almacenada en los diferentes TCE's de líneas.

En las transacciones de alto nivel, la FMM que las solicita no se limita, como en el caso anterior, a enviar una sentencia DML solicitando el cambio, sino que debe tener mayor control. Esta FMM debe conocer los CE's involucrados y mantener un diálogo con los DBCS's de estos CE's.

En la figura de la página 147 se han representado los mensajes a intercambiar. Analicemos esta figura:

TRANSACCIONES DE ALTO NIVEL



1. El programa de aplicación informa al DBCS del comienzo de la transacción.
2. El DBCS anota el comienzo (SOT) en el fichero de seguridad.
3. Se pide el primer cambio al procesador B.
8. Idéntica secuencia para cambio de dato simple.
9. El programa de aplicación recibe el reconocimiento.
10. Petición del segundo cambio al procesador C.

1.15 Idéntica secuencia para cambiar un dato.

1. La FMM que controla la transacción informa al DBSS de que va a comenzar una.
2. El DBSS abre en el disco un fichero de seguridad para ir almacenando en él los diversos pasos de que está compuesta la transacción anotando, de momento su arranque (SOT= State of Transition).
3. La FMM pide al DBCS del primer CE involucrado que proceda a cambiar. Esta petición la realiza a través de un conando DML y el mecanismo arrancado es el mismo al visto en la figura de la página 147 del 3. al 9. y del 10 al 16 idéntico para otro CE y así sucesivamente hasta acabar la transacción. En este momento hay que proceder a terminarla.

17. La FMM informa al DBSS que ha acabado.
18. El DBSS cierra el fichero de la transacción (EOT= fin de transacción).

Mecanismos de Seguridad.

Un objetivo del DBSS es mantener la consistencia en los datos. Para ello están previstos tres tipos diferentes de mecanismos, alguno de ellos, ya implícitamente visto en la explicación anterior:

a. Recuperación por Vuelta a Atrás.

b. Recuperación desde el Estado Inicial.

c. Auditorías.

a. Recuperación por Vuelta a Atrás. (ROLL-BACK)

Gracias a haber abierto un fichero de seguridad. Si una vez empezado un cambio éste, por algún motivo, no se puede terminar con éxito, el DBSS obtiene del fichero de seguridad el valor antiguo cargando en los datos este valor.

En la figura de la página 150 se ha representado este mecanismo.

b. Recuperación desde el estado Inicial. (ROLL-FORWARD).

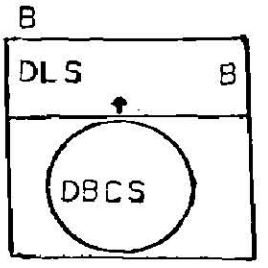
Si en un momento en que se está procediendo a cambiar los diferentes DLS's, se encuentra un fallo, dudándose de la fiabilidad del dato en el disco, es posible volver a sacar de la cinta los DLS originales. Una vez cargados en el disco, estos DLS, se actualizan con los nuevos datos obtenidos.

En la figura de la página 151 se ha representado este mecanismo.

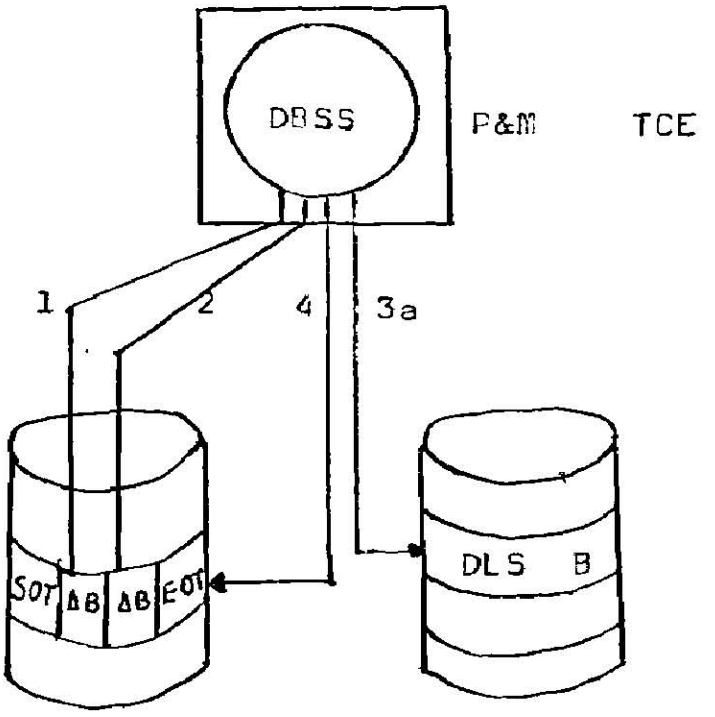
c. Auditorías.

La función de éstas es asegurar que todas las co-

TRANSACCION HACIA ATRAS (ROLL BACK)



3b

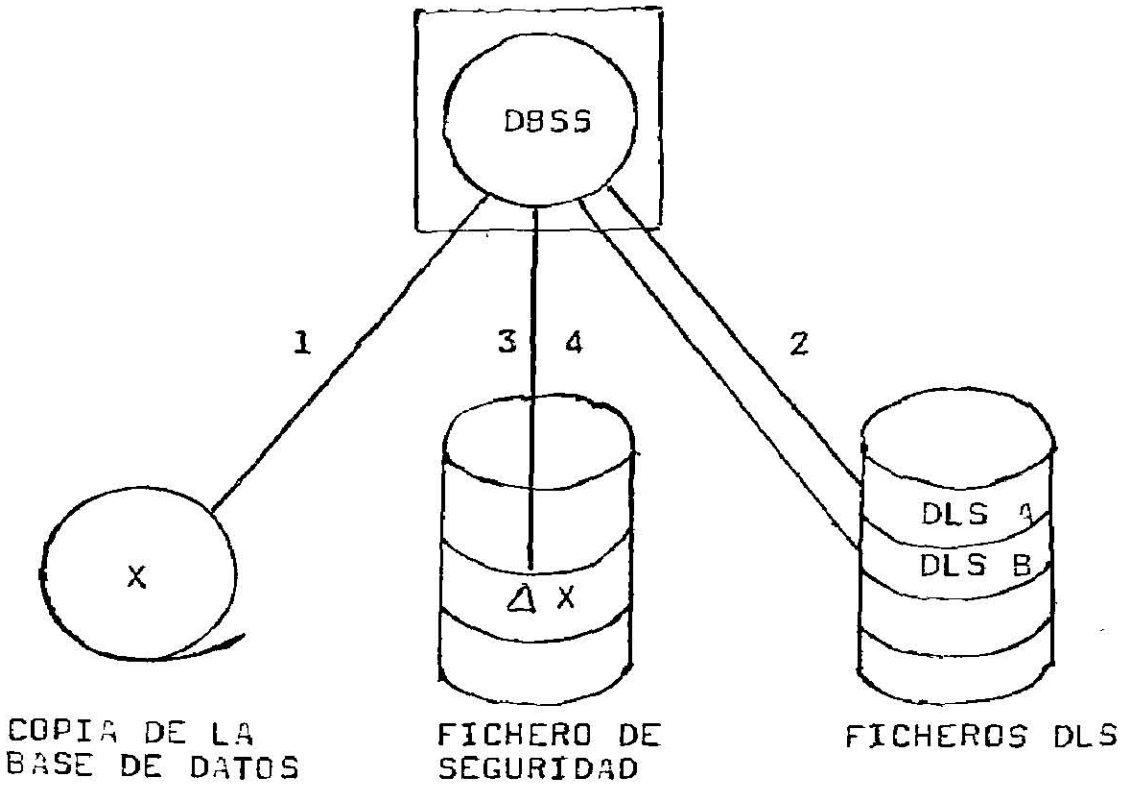


1. La función del Roll Back (DBSS) lee el cambio de la transacción que tiene que ser "vuelta hacia atrás".
2. El cambio se anota de nuevo en el fichero de seguridad.

3a y 3b. El DLS adecuado es devuelto a su valor antiguo, tanto en el disco como en la memoria.

4. Se cierra la transacción abierta.

VUELTA HACIA ADELANTE (ROLL FORWARD)



1. Desde la cinta se lee la copia antigua de la base de datos.
2. Se actualiza la base de datos en el disco.
3. Se obtienen los cambios de los ficheros de seguridad.
4. Se implementan los cambios en la B.D.

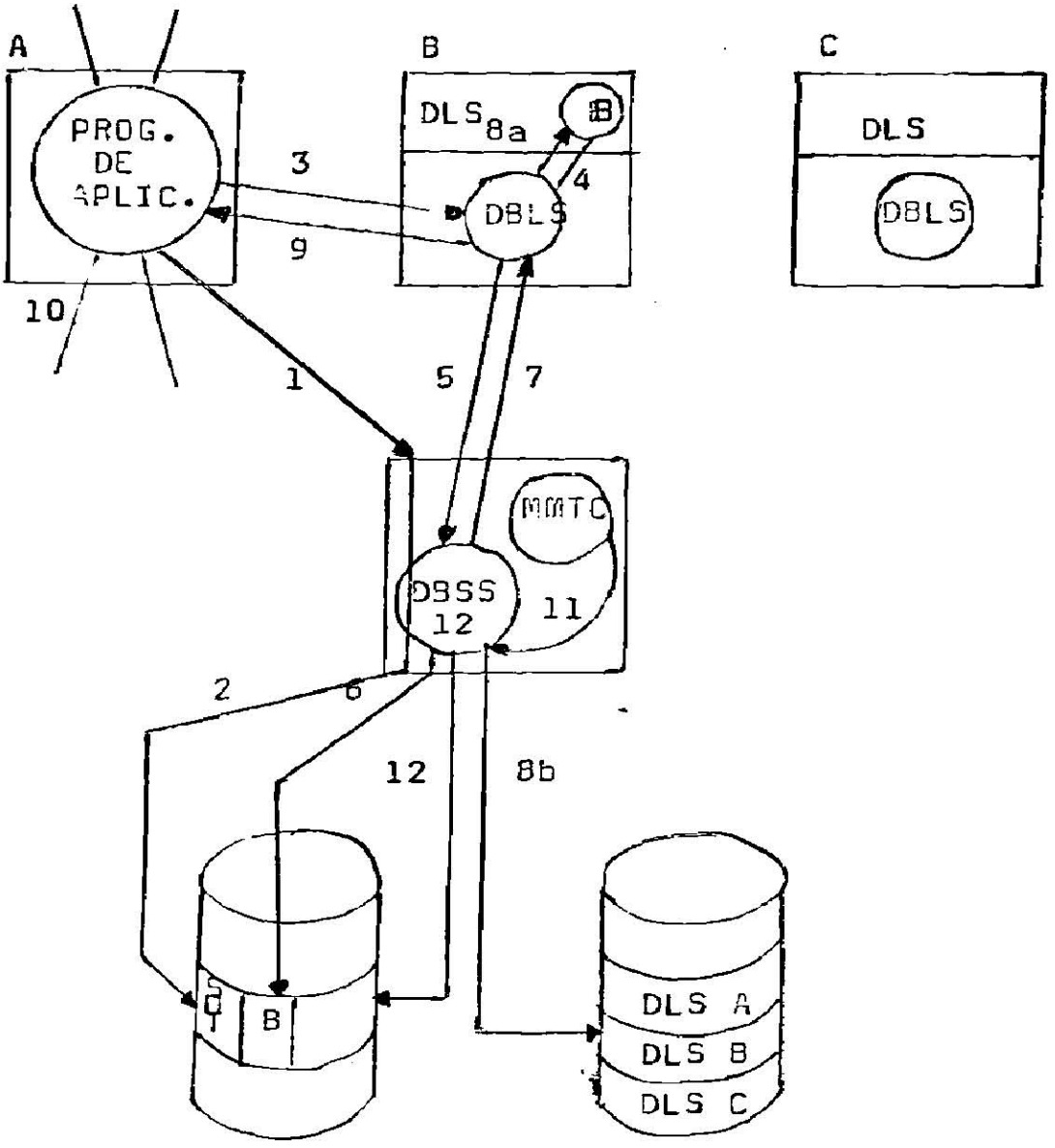
pias de una relación replicada y almacenada en diferentes CE's tienen el mismo contenido.

Las inconsistencias (distintos valores en las diferentes representaciones) pueden haberse creado por fallos del sistema, coro por ejemplo la caída de un procesador durante una operación de actualización.

Cuando el procesador llega a estar de nuevo operacional y se arranca la adecuada rutina de recuperación, se arrancarán las auditorías y se detectarán estas inconsistencias.

En la figura de la página 153 se ha representado la situación en la que está ejecutándose una transacción de alto nivel, a la mitad de la misma ha caído el CE de la FMM que controla la ejecución. Cuando éste está de nuevo operacional, mantenimiento pedirá una auditoría y ésta encontrará una transacción no acabada (gracias al fichero abierto para controlar la transacción).

FALLO EN TRANSACCION DE ALTO NIVEL



1. El programa de aplicación informa del corienzo de la transacción.

1.9 Secuencia idéntica a la de la figura de la página 147 .

10. Caída del procesador donde estaba ejecutándose el programa de aplicación.

11. Mantenimiento pide una auditoría.

12. Auditoría encuentra una transacción no acabada y pide que entre la función de recuperación.

B I B L I O G R A F I A

Arras, J. and H. Tarle. P C M Line Equipment ZAD 2. Ericson Review, 49, 1972:2.

Cattermole, K.W. Principles of Pulse Code Modulation. Iliffe, London 1969.

CCITT GREEN BOOK. Digital Transmission Systems. Volume III-2 Part I, Section 7.

ERICSON REVIEW NUM. 4 1978.

Geissler H. Planning of a PCM System Hierarchy. NTZ Report 8, VDE- Verlag Gmbh, Berlín, 1971.

INDETEL. Manual Capacitación y Desarrollo. Sistema PCM. México, D.F. 1981.

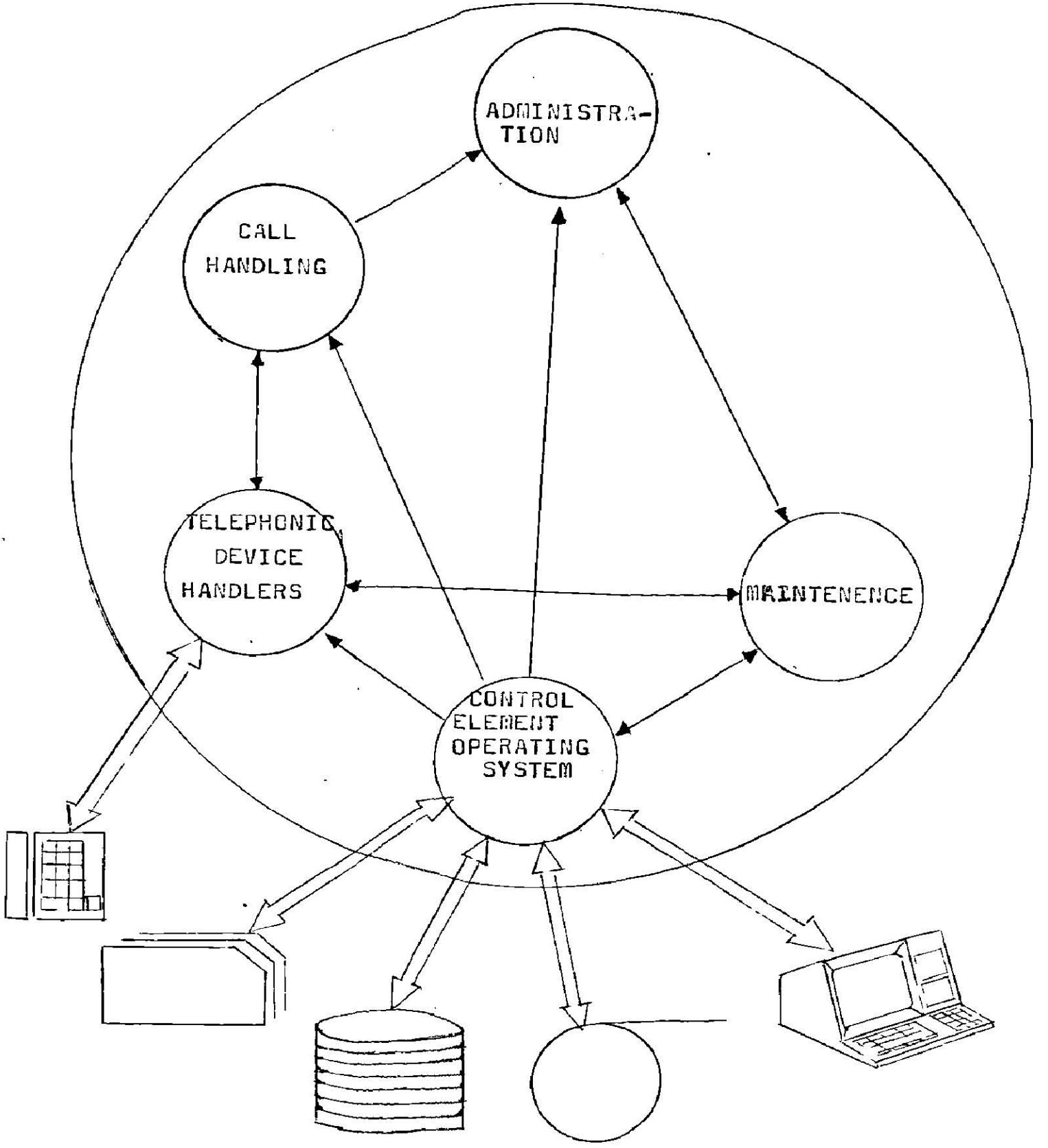
Jacobowitz, Henry. Computadoras Electrónicas. Compañía General de Ediciones, S.A., Sexta Edición, México, D.F., 1979.

KÜnding, A. Digitale Telephonie. Blau TR-Reihe Heft 103, Verlag. Technische Rundschau. Hallwag, Bern und Stuttgart, 1972.

Lindquist, et. al., S. P C M Multiplexing Equipment. ZAK 30/32, Ericson Review, 49, 1972:2.

Orellana, Roberto H. Ing. Telefonía Elemental. Teléfonos de México, S.A., Escuela Tecnológica. México, D.F., Mayo 1959.

Widl, W. PCM Transmission. Ericson Review, 49, 1972.



Software Functional Structure