PATENTE DE INVENCION

Case 4. Div.

Int. Cl.: G06F

434287

Memoria Descriptiva

sobre:

Perfeccionamientos en dispositivos procesadores de datos.

- - - - - - - - - - -

Solicitante: BURROUGHS CORPORATION, entidad norteamericana, residente en Burroughs Place, Detroit, Michigan 48232, EE. UU. de A.

- - - - - - - - - - -

Ordenador digital electrónico, serial para caracteres, que utiliza un vocabulario de cuatro caracteres, uno de cuyos caracteres está representado por dos bitios binarios, que se extructura para procesar datos seriales por caracter que llegan al ordenador de una manera es-

M - 2

pecifica y se inician por los datos de llegada. Las estructuras de los datos que pueden representar programa u operaciones a realizar sobre los datos que llegan a la entrada del ordenador se almacenan en el área de almacenamiento del ordenador en forma de estructuras de datos agrupados que se pueden ilustrar como tres estructuras donde cada nodo de las tres estructuras representa una operación, Las estructuras de los datos que pueden representar operandos se abastecen también al ordenador en una organización agrupada. Este dato operando localiza un cierto nodo u operación residente en el área de almacenamiento del ordenador. El enlace del dato operando de llegada con su dato de programa excita la ejecución de la operación. En el caso de que se necesiten más de un operando antes de que pueda realizarse una operación, la llegada de un primer operando sin el segundo causa almacenamiento en el primer operando hasta la llegada del segundo. La llegada del segundo operando excita el comienzo de la operación. Esta interrelación de datos de programa y datos operandos, o sea, el enlace del dato dinámico con el dato estático para excitar la operación existe tanto si el dato del progrma está almacenado y es estático como si el dato operando se almacena y es estático. Utilizando un vocabulario de cuatro carácteres para representar los datos, utilizandose dos de los carácteres para indicar el comienzo y el final de un campo de datos, se facilita la ejecución de una técnica de comprobación de errores según la cual solamente se cuentan los caracteres detectados indicativos del comienzo y el final de un campo de datos. La utilización del comienzo y el final de caracteres del campo de datos en las estructuras de los datos consistentes en campos de datos agrupados permite a voluntad la expansión

y contracción de los campos en el mismo.

El presente invento se refiere en general a perfeccionamientos en elaboradores de datos digitales y, de un modo más particular, se refiere a sistemas elaboradores de datos digitales nuevos y perfeccionados que se caracterizan porque el elaborador de datos es un dispositivo de circuito integrado microprogramado.

En el campo del proceso de datos digitales, actualmente es una práctica común emplear arquitecturas de sistema que se han desarrollado pero a costa de un elevado gasto en componentes. Esta restrincción ha dado por resultado la centralización de control de sistema en dispositivos conocidos como elaborador central y unidades de memoria principal. Debido a éste sistema centralizado masivo y costoso que exigía control, se desarrollaron sistemas operatorios(programas de control maestro) para generalizar su utilización, compartiéndolo a través de un cierto número de programas o áreas. Las arquitecturas de los sistemas que se obtienen por estas influencias son de una gran generalización y, como resultado, son innecesariamente complejas, ad hoc, e ineficaces con respecto a un gran número de situaciones particulares. Este tipo de arquitectura se divide de una manera irregular y se pone en práctica principalmente mediante una lógica secuencia del juego fijo. Cuando se utilizan técnicas de microprogramación, la arquitectura funcional básica del sistema no cambia en el sentido de que los elaboradores microcodificados siguen todavía arquitecturas secuenciales sincronizadas orientadas con registradores.

Las nuevas tecnologías de circuitos integrados, por ejemplo de los tipos MSI y LSI que proporcionan los elementos

esenciales de un elaborador de datos en un solo bloquecito pueden utilizarse con eficacia solamente si se sigue un nuevo juego o conjunto de compulsiones o compromisos de diseño. La tecnología de LSI, por ejemplo, exige irregularidad de componentes y la no dedicación de algoritmos especializados o complejos en los bloquecitos de circuito. Adicionalmente, como las memorias de circuitos integrados son compatibles en interfase con la lógica de los circuitos integrados, se puede eliminar el esquema de la arquitectura de elaborador orientado con registradores distribuyendo la memoria del circuito del sistema a través de dicho sistema. Esto, lógicamente, elimina la necesidad del empleo de un subsistema de memoria principal centralizado. Actualmente es factible distribuir memoria de sistema por todo un sistema, por lo que es conveniente eliminar los sistemas operatorios de control central necesarios con anterioridad a éste invento.

Para poder utilizar con eficácia la tecnología de LSI, se requiere una arquitectura de sistema que de por resultado un sistema divisible bien formado y regular. Aunque casi todas las técnicas de microprogramación utilizados anteriormente persiguen este objetivo, las técnicas de programación anteriores no han podido producir un sistema eficaz de programar y eficaz en la ejecución de sus algoritmos. En otras palabras, estos sistemas microprogramados de la tecnología anterior tienen una total carencia de continuidad entre lo que es el lenguaje de la máquina y lo que son las necesidades de programación del usuario y demanda del lenguaje. Estos se debe a que los lenguajes de microcódigos de máquina de la tecnología anterior son por naturaleza seriales y enlazantes al contrario que la tecnología de LSI que demanda regularidad pe

ro no ligazón de funciones complejas.

Este invento tiene por objeto proporcionar un elaborador digital que puede utilizarse como bloque de construcción básica, por ejemplo un ordenador de elaboradores múltiples que no necesita utilizar un programa de control maestro ni exige un sistema de interrupción extenso y que tiene capacidades de emulación mejoradas.

Los problemas que llevan consigo los sistemas centrales y otros inconvenientes mencionados anteriormente, se resuelven en este caso y se consigue los objetos mencionados mediante un sistema de datos binarios que se caracteriza porque el almacenamiento contiene archivos de datos compuestos por campos de datos y carácteres de datos y la circuiteria del sistema tiene circuítos para recibirlos. Un campo de datos contiene la dirección o localización del archivo del dato de almacenamiento.

El sistema puede describirse en general, según lo llamaremos como un sistema activado por datos.

La finalidad general de éste invento se consigue de un modo más particular utilizando un vocabulario de caracteres múltiples en un elaborador de datos serial por caracteres que se caracteriza porque dos de los carácteres se utilizan para definir el comienzo y el final de un campo de dato particular. Cada carácter está representado por una pluralidad de bítios binarios. Las estructuras de los datos se organizan en archivos de datos que contienen campos de manera que permita la expansión y contracción de estos campos. Cada campo de dato termina preferiblemente en un código de final de campo que excita una comparación entre el contaje de caracteres de principio y final de campo en la estructura del da-

to y un contaje de referencia. La estructura y organización de un archivo se describe por el contenido del primer campo en dicho archivo. Un programa o proceso se lleva a cabo en respuesta de la unión de pares de archivos de datos, teniendo cada par un archivo de datos que contienen una parte del programa y conteniendo el otro archivo de datos los operandos para dicha parte del programa. Uno u otro tipo de archivo de datos puede residir en el área de almacenamiento del elaborador de datos (estática)mientras que el otro se alimenta al elaborador del exterior (dinámico). La llegada de los archivos de datos dinámicos hace que se localice el archivo de datos coincidentes en almacenamiento. A su vez, el archivo vector puede dar lugar a la ejecución de la operación dictada por su contenido utilizando los operandos abastecidos por los archivos de operandos entrantes. Si todos los operandos para la estructura del dato localizado están presentes o han llegado se efectúa la operación designada por la estructura del dato del programa, transmitiéndose el resultado a un destino indicado por la estructura del dato del programa. Los dos archivos de dato coincidentes pueden utilizarse en combinación para producir la resultante dictada por el archivo de datos del programa.

Otros objetos y muchas de las ventajas consiguientes de éste invento se comprenderán con facilidad en el transcurso de la descripción detallada que sigue, tomando como referencia los dibujos adjuntos, donde los números iguales de referencia indican partes componentes iguales en todas sus figuras, y donde:

La figura 1 es una ilustración esquemática de conjuntos de un sistema de proceso de datos de elaborador simple, se

gún el invento.

La figura 2 es un diagrama lógico de la fila de espera de entrada en el procesador de la figura 1.

La figura 3 es un diagrama lógico de la unidad lógica vectorial en el procesador de la figura 1.

La figura 4 es un diagrama lógico de la unidad de control del procesador de la figura 1.

La figura 5 es un diagrama lógico de la fila de espera de salida del procesador de la figura 1.

La figura 6 es un circuito lógico de un circuito de reconocimiento de señales utilizando en la fila de espera de entrada de la figura 2.

La figura 7, es una ilustración abstracta que representa en forma de árbol un ejemplo particular de un programa que puede ser ejecutado por el ordenador de la figura 1.

La figura 8 es una ilustración abstracta de un algoritmo simple representado en forma de árbol y la estructura del dato o archivo que representa dicho algoritmo que es utilizado por el procesador de la figura 1 para realizar las operaciones especificadas.

La figura 1 ilustra un sistema de un procesador activado por datos 11 en comunicación con una pluralidad de unidades periféricas 15,17, 19 a través de un intercambiador de entrada/salida 13. El intercambiador de entrada/salida 13 puede ser un tipo normal de circuito de conmutación, como el que se utiliza en centrales telefónicas, donde cualquiera de las unidades periféricas puede conectarse al procesador activado por datos 11 por medio de un cable de entrada 31 o un cable de salida 33. Las unidades periféricas pueden ser unidades de formato en paralelo o en serie. Para que sirve la naturale

za serial por carácteres del elaborador 11 cuando se utilizan unidades de formato en paralelo, el intercambiador de entrada/salida 13 comprendería un multiplexor para convertir la pluralidad de trayectos de señales paralelos procedentes de las unidades periféricas 15,17, 19, a la entrada de trayecto de señales relativamente serial al elaborador 11. Para acomodar la transmisión de señales seriales por carácteres dede el elaborador 11 hasta las unidades periféricas de formato en paralelo 15 a 19, el intercambiador de entrada/salida 13 comprendería un desmultiplexor. Las unidades periféricas 15,17, 19 pueden ser cualquiera de los aparatos bien conocidos tales como aparatos de cinta magnética, lectores de tarjetas, aparatos perforadores de tarjetas, unidades de teclado, impresora, o dispositivos de almacenamiento de tambor o disco.

El ordenador digital activado por datos o procesador de datos 11 recibe estructuras de datos de las unidades periféricas en su fila de espera de entrada 21. Estas estructuras de datos, según se explicará más adelante, tienen una organización especializada y deben seguir ciertas reglas de síntaxis. La fila de espera de entrada 21 es básicamente una unidad FIFO (unidad tampón de prioridad de salida por prioridad de entrada) que realiza la función adicional de sincronizar las estructuras de datos asincronos recibidos en el cable de entrada 31 al cronometrador del sistema del ordenador. Las estructuras de datos recibidas por la fila de espera de entrada 21 se reciben de una forma serial por carácteres. Estas estructuras de datos pueden considerarse como comunicadas a los demás elementos del elaborador 11 de una manera serial por carácteres. La estructuras de los datos en la fila de espera de entrada 21 se transmiten a almacenamiento del ordenador 25,

por ejemplo de una manera serial por caracteres sobre el cable 35, hasta una unidad de control 23, y desde la unidad de control 23, por el cable 51, al almacenamiento del ordenador 25 La comunicación de control de la fila de espera de entrada 21 y la unidad de control 23, por el cable 37, y la comunicación de control entre el control 23 y el almacenamiento o memoria 25, por el cable 49, se explicará más adelante.

Además de las estructuras de datos procedentes de la fila de espera de entrada 21 que se transmiten al almacenamiento 25, se pueden transmitir a una unidad lógica vectorial 27 por medio del control 23 por el cable 47. De igual modo, las estructuras de los datos procedentes de almacenamiento 25 pueden comunicarse a la unidad lógica vectorial 27 por medio de la unidad de control 23 por el cable 45. La comunicación de control entre la unidad lógica vectorial 27 y la unidad de control 23, por el cable 43 se explicará más adelante.

La unidad lógica vectorial 27 es básicamente una unidad aritmetica serial que realiza, por ejemplo, funciones básicas tales como suma, resta, comparación y envio sobre estructuras de datos de longitud de campo variable. La unidad lógica vectorial se puede comunicar directamente con almacenamiento 25, por el cable de datos 53, con una fila de espera de salida 29, por el cable de datos 59. La comunicación de control entre la unidad lógica vectorial 27 y almacenamiento 25, por el cable de control 55, y con la fila de espera desalida 29, por el cable de control 57, se explicará más adelante.

El almacenamiento o memoria 25 del ordenador activado por datos 11 puede ser una memoria de circuitos integrados de acceso aleatorio de un tamaño preferible construído

de bloquecitos de memoria de acceso aleatorio. La construcción de una memoria de tamaño mayor con dicho bloquecito de memoria se considera dentro de los conocimientos normales del experto en la materia. Otro objeto del bloquecito de memoria que puede utilizarse para construir la memoria 25 del ordenador puede ser un bloquecito de memoria de contenido localizable de gran velocidad.

La fila de espera de salida 29 que puede recibir estructuras de datos de la unidad lógica vectorial 27, almacenamiento o memoria 25, o la fila de espera de entrada 21, realiza la función de colocar las estructuras de los datos que ha recibido de forma que se transmiten a las unidades periféricas 15-19 por medio del intercambiador de entrada/salida.13. La fila de espera de salida, al igual que la fila de espera de la entrada, es básicamente del tipo de una unidad tampón FIFO que acepta estructuras de datos de una manera serial por caracteres y transmitir estos carácteres al intercambiador de entrada/salida.

Refiriéndonos ahora, a la figura 2, la fila de espera de entrada 21 se comunica con el intercambiador de entrada/salida por el cable 31. El cable 31 está compuesto por líneas 79,81,83 y 85 que salen del circuito lógico de interfase 61 en la fila de espera de entrada 21 o se dirigen al mismo. Las líneas 85 son dos líneas de datos paralelas que reciben dos bitios en paralelo desde el intercambiador de entrada/salida (figura 1). Estos dos bitios paralelos representan un carácter. Las otras tres líneas 79,81 y 83 son líneas de control en la fila de espera de entrada y el intercambiador de entrada/salida. La línea 79 transmite un nivel de señales binarias que dan instrucciones al intercambiador de entrada/salida para

que transmita la estructura del dato siempre que se detecte un error en la estructura del dato recibido con anterioridad. La línea 81 lleva un nivel de señales binarias que activa o desactiva el intercambiador de entrada/salida con respecto a la transmisión de estructura de datos. La línea 83 lleva un nivel de señales generado por el intercambiador de entrada/salida que indica una petición para enviar estructuras de datos desde una de las unidades periféricas o la fila de espera de salida del elaborador de datos 11. En respuesta a dicho nivel de señal de petición el nivel de señal en la línea 81 activaría al intercambiador de entrada/salida si la fila de espera de entrada pudiera contener datos adicionales.

La estructura de datos seriales por caracteres recibida en las líneas 85 desde el intercambiador entrada/salida 13 (figura 1), además de remitirse al circuito lógico de interfase 61, se comprueba para hallar errores por parte de la circuiteria lógica, llamada por conveniencia circuito lógico de reconomiento de "paréntesis", y un contador ascendente/descendente binario 65 que responde a la circuiteria de reconocimiento "parentesis" 63. El computo del contador 65 se transmite al circuito lógico de interfase 61 por el cable 93. En este punto es suficiente indicar que si el computo del contador binario ascendente/descendente 65 al final de una estructura de dato particular no es cero, el circuito lógico de interfase 61 pide una retransmisión por la línea 79 porque ha ocurrido un error en la estructura de dato. La lógica especifica del circuito de reconocimiento de "paréntesis" 63 y su interacción con el contador ascendente/descendente 65 y el circuito lógico de interfase 61 se explicarán con más detalle más adelante.

Según se ha indicado anteriormente. La fila de espera de entrada 21 funciona básicamente con una unidad tampón FIFO y sincrona los caracteres de datos entrantes asincronos con el cronometrador del sistema del ordenador (no ilustrado) que forma parte del circuito lógico de interfase 61. La parte tampón de la fila de espera de entrada es la memoria de fila de espera de entrada 67 que puede ser una memoria de acceso aleatorio construída a partir del bloquecito de memoria de acceso aleatorio de circuito integrado.

Los caracteres de los datos recibidos en la línea 85 desde las unidades periféricas se transmiten a la memoria de fila de espera de entrada 67 por las líneas 96, donde se almacenan en el espacio disponible siguiente según indica el circuito indicador de escritura 73. Entre el almacenamiento de caracteres de datos en la memoria de fila de espera de entrada, se toma la lectura de los caracteres de los datos de esta memoria y se transmite a los demás componentes del elaborador 11 (figura 1), por medio de la unidad de control 23 (figura 1). El caracter de dato particular que se lee en la memoria 67, en cierto instante, esta determinado por el circuito indicador de lectura 71. El caracter del dato que se lee en la memoria de fila de espera de entrada se transmite desde la memoria de fila de espera de entrada por las líneas 98 hasta el circuito lógico de interfase 61 y despues a la unidad de control 23 (figura 1) por las líneas 35. Las líneas de control 123,121, que componen el cable de control 37, llevan señales de activación de lectura y de petición de lectura desde la unidad de control 23 (figura 1). La línea 123 lleva una señal de activación de lectura. La línea 121 lleva una señal de peti-

ción de lectura. En términos generales, entonces se almacena información en la memoria de la fila de espera de entrada 67 tan pronto como se recibe y se lee de la memoria de la fila de espera de entrada 67 en un orden de preferencia de entrada tan pronto como la unidad de control 23(figura 1) lo pide. Cuando el circuito lógico de interfase 61 recibe caracteres de datos por las líneas 85, genera una señal en la línea 97 hasta una unidad de control de ciclo de memoria 69 indicando que se requiere una función de escritura. El control del ciclo de memoria, en respuesta a ésta petición de escritura genera una señal de activación de escritura, en la línea 103, a la memoria de la fila de espera de entrada 67, una señal de selección de escritura, en la línea 105 a un selector 75, y una señal de incremento en la línea 99, a un indicador de escritura 73.

Básicamente, el selector, en respuesta a una señal de selección de escritura o lectura en la línea 105 elige la señal de salida del indicador de escritura o lectura alimentada al mismo por el cable 109 y 111, respectivamente, para transmitir por el cable 107 al registrador de direcciones 67 de la memoria de la fila de espera de entrada 67.

El indicador de escritura 73 y el indicador de lectura 71 pueden ser un contador binario. Las entradas de incremento 99 y 101 al indicador de escritura y al indicador de lectura, respectivamente, procedentes del control del ciclo de memoria 69 se conectaría a la entrada A (no ilustrada) de estos contadores Signetics. La línea 100 procedente del circuito lógico de interfase 61 al indicador de lectura 71 y al indicador de escritura 73 se conectarían a las entradas de reposición (no ilustrado) de estos contadores.

La salida del indicador de escritura y el indicador de lectura además de pasar por el selector para localizar la memoria de la fila de espera de entrada 67, se muestrean por medio de un comparador 77. Este comparador tiene dos conductores de salida que indican cual de las dos entradas es mayor y cuando son iguales. Como la fila de espera de entrada 67 funciona con prioridad de salida de acuerdo con la prioridad de entrada, o sea, como una unidad tampón de prioridad de salida de acuerdo con la llegada, es contaje del indicador de escritura será siempre mayor que el contaje del indicador de lectura, siempre que la memoria de la fila de espera de entrada 67 tenga contenidos datos pero no esté llena. Por lo tanto, una señal en la línea 119 procedente del comparador 77 indicará al circuito lógico de interfase 61 que el cómputo de indicador de escritura es mayor que el contaje del indicador de lectura. Esto indica al circuito lógico de interfase que el dato permanece todavía en la memoria de la fila de espera de entrada.

Siempre que el contaje del indicador de escritura es igual al contaje del indicador de lectura, se transmite una señal desde el comparador, por la línea 117, hasta el circuito lógico de interfase 61. Esta señal puede indicar que la memoria de la fila de espera de entrada 67 está completamente vacia o completamente llena, dependiendo de si la última solicitud de la memoria generada por el circuito lógico interfase 61 interpreta la señal en la línea 117 como indicativa de que la memoria de la fila de espera de entrada 67 está llena si la última operación de la memoria fue una operación de escritura. Si la última operación de la memoria fue una operación de lectura, se toma una señal en la línea 117 como indicación

de que la memoria de la fila de espera de entrada está vacia. El circuito lógico de interfase 61 sabe si la última operación de la memoria fue una operación de escritura o de lectura puesto que transmite una solicitud de escritura o de lectura por las líneas 97,95 respectivamente, al control del ciclo de la memoria 69. Siempre que el circuito lógico de interfase 61 determina que la memoria de la fila de espera de entradas 67 está vacía, genera una señal de reposición en la línea 100 que se alimenta a ambos indicadores de escritura y de lectura.

La circuiteria lógica especifica del control del ciclo de la memoria 69 y el circuito lógico de interfase 61 no se describen en la presente porque la ejecución de las funciones atribuidas a estos circuitos lógicos queda perfectamente dentro de los conocimientos del experto en la materia.

Refiriéndonos ahora a la figura 3, una unidad lógica vectorial serial 27 que se puede utilizar en el ordenador de la figura 1, se ilustra consistiendo básicamente en dos ROM (memorias de lectura solamente) 125 y 129. Ambas ROM pueden ser del tipo fabricado por la Signetics Corporation y relacionadas en su catálogo de accesorios del 1972 páginas 4-1. Los registradores de direcciones o localizaciones 124 y 128 para la memoria de lectura solamente 125 y 129, respectivamente, son registradores de direcciones o localizaciones de entrada en paralelo y de salida en paralelo normales. La única diferencia estructural entre las dos memorias de lectura solamente reside en el microcódigo contenido dentro de las mismas. La memoria de lectura solamente 125 contiene el microcódigo necesario para generar los resultados de operaciones dinámicas tales como suma, resta, o comparación, por ejemplo.

Las estructuras de los datos que proceden en serie por caracteres de almacenamiento 25 del ordenador 11 (figura 1) por medio dela unidad de control 23 por las líneas 45 hasta la unidad lógica vectorial 27 se dirigen por el desmultiplexor 135, según sea una señal de control en la línea 43a procedente de la unidad de control 23, a la ROM diádica 125 por la línea 139, o a la ROM monádica 129 por la línea 142, dependiendo de que clase de estructura de dato sea localizada por la estructura de dato en la fila de espera de entrada 29. Esto se explicará más adelante con más detalle.

De igual modo, el desmultiplexor 137 recibe datos seriales por caracteres por las líneas 47 desde la fila de espera de entrada 21, por medio de la unidad de control 23, y las dirige a la ROM diádica 125 por la línea 141 o a la ROM monádica 129 por la línea 143. La salida de la ROM diádica 125 o la ROM mónadica 129 se dirigirá al almacenamiento o memoria 25 del ordenador o a la fila de espera de salida 29 del ordenador (figura 1), dependiendo de la dirección de destino contenida dentro de la estructura de dato del programa. Esta dirección o localización de destino se alimenta a desmultiplexores 133 y 130 por las líneas 43d por la unidad de control 23 del ordenador 11 (figura 1).

Los desmultiplexores 135, 137, 130 y 133 utilizados en esta unidad lógica vectorial pueden ser del tipo fabricado por la Signetics Corporation e ilustrados en su catálogo de accesorios 1972 en la página 2-132.

Suponiendo, a título de ejemplo, que se realice una operación diádica, sumándose un operando A con un operando B, se alimentaría un código OP que designa la operación diádica de adicción al registrador de direcciones 124, bien desde al-

macenamiento o memoria 25 o la fila de espera de entrada 21 del ordenador, por las razones que se evidenciarán mas adelante. Junto con este código OP, los dos operandos se alimentan también, de una forma serial por caracteres, al registrador de direcciones 124. Como resultado, la salida en el cable 126 de la memoria de lectura solamente 125 sería el resultado serial por caracteres de la suma de los dos operandos. Efectivamente, lo que ocurre es que el código OP, además de los operandos, actua como direcciones a las áreas particulares en la memoria de lectura solamente 125 que almacenen los resultados de la suma de dos caracteres particulares de los dos operandos que se suman.

Las salida de la memoria de lectura solamente 125, en este ejemplo particular, contendría también una señal en la línea 43c que indicaría a la unidad de control 23 (figura 1) que se completa una suma de caracteres particulares. Asi mismo, cuando se trata de una suma, se propagan señales portadoras de nuevo a la entrada de la memoria de lectura solamente 125 por las líneas 132 para modificar la suma de los caracteres siguientes. En caso de que se realicen operaciones monádicas son memoria de lectura solamente 129, las líneas de realimentación 131 pueden ser simplemente una entrada de contador progresivo para modificar el contenido del registrador de direcciones 128 del ROM monádico con lo que se localiza el siguiente lugar de la memoria.

En resumen, la unidad de control 23 introduce estructuras de datos desde el almacenamiento o memoria 25 y la fila de espera de entrada 21 hasta la unidad lógica vectorial 27 que responde a estas dos estructuras de dato generando señales de resultado más control, que se vuelven a enviar a al-

macenamiento 25 por las líneas 53 y 55, o a la fila de espera de salida 29 por las líneas 57 y 59.

Tomemos ahora como referencia la figura 4 que ilustra la unidad de control 23 del ordenador 11 como una unidad microprogramada consistente en una pluralidad de memorias de lectura solamente y multiplexores. La ROM analizadora de campos 146 recibe estructura de datos de la fila de espera de entrada, por las líneas 35, o de almacenamiento, por las líneas 51b. La estructura del dato procedente de la fila de espera de entrada 21 (figura 1) o la estructura del dato procedente de almacenamiento 25 (figura 1) localiza la ROM analizadora de campos 146 a través del registrador de direcciones o localizaciones 145 haciendo que la ROM analizadora del campo 146 responda enviando señales de control a uno de la pluralidad de los desmultiplexores 148, 150 y 152.

Por ejemplo, si la estructura del dato procedente por la línea 35 de la fila de espera de entrada (figura 1) fuera un archivo de operandos, el analizador de campos dirigiria al desmultiplexor 148 para transmitir los campos de operandos por una de las tres líneas 47a, 39a, o 51a, llevando la línea 47a hasta la unidad lógica vectorial, la línea 39a hasta la fila de espera de salida, y la línea 51a hasta el almacenamiento o memoria del ordenador. El analizador de campos, en este caso, respondería al campo de descripción en el archivo de operandos. De igual modo, si una estructura de dato que llega por la línea 51b desde almacenamiento (figura 1) fuera un archivo o campo de operandos, la ROM analizadora de campos 146 dirigiria al desmultiplexor 152 por la línea 162 para que transfiriera el dato por la línea 39b o la línea 45, cuya línea 39b lleva hasta la

fila de espera de salida y la línea 45 lleva hasta la unidad lógica vectorial.

Supóngamos ahora que en lugar de una estructura de dato del operando recibida por las líneas 35 o 51b, se recibiera una estructura de datos de programa. Esta estructura de dato de programa localizaría la ROM analizadora de campo 146 haciendo que transmitiera una dirección a una de las ROM 154, 156, 158, por medio de un desmultiplexor 150. Las ROM 154,156 158 componen una libreria de microprogramas que contiene microprogramas particulares. Estos microprogramas son localizados por la estructura del dato que llega por una de las líneas de datos 35 o 51b. Suponiendo que la estructura del dato recibido por la ROM analizadora de campo 146 comienze con un campo que indica que lo que sigue es un archivo de programa, al analizador de campo generaría una pluralidad de señales al desmultiplexor 150 que encaminaría las señales a la ROM de archivo de programa 154, por ejemplo. En respuesta a estas señales que localizan área particulares en esta ROM, se generan señales de control, por las líneas 43, a la unidad lógica vectorial (figura 3) por la línea 41b, a la fila de espera de salida (figura 5) por la línea 121, al circuito lógico de interfase de la fila de espera de entrada (figura 2) y cuando es apropiado por la línea 144, al registrador de direcciones 145 indicando que se completa la operación particular.

Además de recibir estructuras de datos por las líneas 35 y 51b el registrador de direcciones o localizaciones 145 recibe varias señales de control. Por ejemplo, por la línea 123 se alimenta una señal de control de activación de lectura desde el circuito lógico de interfase de la fila de espera de entrada (figura 2). Por la línea 43c se alimenta una se-

ñal de operación completa desde la unidad lógica vectorial (figura 3). Por la línea 4la, la fila de espera de salida (figura 5) suministra una señal de retención que instruye al control que está llena. También se alimenta una señal de continuación al registrador de direcciones o localizaciones 145 desde la librería de ROM por la línea 144.

El registrador de direcciones o localizaciones 145 es un registrador normal de entrada en paralelo y salida en paralelo bien conocido por los expertos en la materia. La ROM analizadora de campo 146 puede ser del tipo fabricado por las SIGNETICS CORPORATION y relacionado en su catálogo de 1972 página 4-1. Las ROM de la libreria de microprogramas 154,156 y 158 pueden ser del mismo tipo. Los desmultiplexores 148 y 142 pueden ser del tipo fabricado por la Signetics Corporation y relacionados en su cátalogo de accesorio de 1.972, páginas 2-132. El desmultiplexor 150 puede consistir en una pluralidad de desmultiplexores en cascada siendo los desmultiplexores individuales del tipo fabricado por la Signetics Corporation y relacionados en su cátalogo de accesorios de 1972 en las páginas 2-130.

Refiriéndonos ahora a la figura 5, la fila de espera de salida 29 se ilustra como un circuito FIFO de doble memoria. El circuito de control de entrada 145 recibe datos desde la fila de espera de entrada o la memoria o almacenamiento por las líneas 39 por medio de la unidad de control 23 (Figura 1) Las líneas 41 llevan señales de control desde la unidad de control 23 (figura 1). El circuito de control de entrada 145 recibe también datos desde la unidad lógica vectorial 27 por las líneas 59 y, similarmente, transmite y recibe control desde la unidad lógica vectorial 27 por las líneas 57. El dato

recibido por el control de entrada 145 por las líneas 39 se encamina bien a la memoria 155 de operadores RAM (memoria de acceso aleatorio o a la memoria 157 de dirección o de destino RAM dependiendo de si la estructura del dato recibido es una dirección de destino, según determine las señales en la línea de control 41 procedente de la unidad de control 23 (figura 1), o es un operando, según determine las señales en la línea de control 41. El dato recibido en las líneas 59 por el control de entrada 145 se encamina a la memoria de operandos o la memoria de direcciones de destino, según determine las señales en las líneas de control 57.

Tanto la memoria de operadores como la memoria de direcciones de destino pueden estar compuestas por bloquecitos de memoria RAM fabricados por la Signetics Corporation e indicados en su catálogo de accesorios de 1972 en las páginas 4-20. Ambas memorias son localizadas por un indicador de escritura o un indicador de lectura, teniendo la memoria de operadores 155 un indicador de escritura 147 y un indicador de lectura 163; la memoria de direcciones de destino 157 tiene un indicador de escritura 149 y un indicador de lectura 161, La operación de estos indicadores de escritura y de lectura respectivos es idéntica a la operación que realizan en la fila de espera de entrada cuando localizan la memoria de la fila de espera de entrada 67 (figura 2).

El circuito de control de entrada 145 funciona como el circuito lógico de interfase 61 en la fila de espera de entrada (figura 2) al responder a las señales procedentes de los comparadores 151 y 153 para detener la transmisión de información a la fila de espera de salida 29 desde la fila de espera de entrada, almacenamiento, o unidad lógica vectorial,

Los comparadores 151 y 153, respectivamente, indican al circuito de control de entrada 145, de la misma manera que el comparador 77 de la fila de espera de entrada de la figura 2, quelas memorias respectivas están llenas, vacias o contienen algunos datos.

El circuito de control de salida 159 de la fila de espera de salida 29 inicia una solicitud de lectura desde la memoria de operandos o desde la memoria de dirección de destino RAM 155, 157, respectivamente, en respuesta a recibir una instrucción de transmisión desde el intercambiador de entrada/salida (figura 1) por la línea 167 del cable 33. El control de salida 159 responde también a una señal de transmisión por la línea 165. En respuesta a las señales en una de esta líneas, el circuito de control de salida 159 puede transmitir una solicitud para escribir una señal de dato en la línea 169 o el intercambiador de entrada/salida. Al recibir una señal de transmisión por la línea 167, por ejemplo, la estructura del dato, parte de la cual se encuentra en ambas memorias, se transmite en serie por caracteres por las líneas 171 al intercambiador de entrada/salida 13 (figura 1). Se recordará que el intercambiador de entrada/salida 13 de la figura 1, en respuesta a recibir estructuras de datos por las líneas 171 desde la fila de espera de salida 29, encaminará dichas estructuras de datos de acuerdo con el campo de dirección recibido desde la memoria RAM de localización de destino 157. Así, la unidad periférica 1,2 6 N (figura 1) puede recibir el dato o la estructura del dato puede encaminarse directamente a la fila de espera de entrada del ordenador 11, para elaboración adicional.

Refiriéndonos ahora a la figura 6, se ilustra la lógi-

ca especifica del circuito de reconocimiento de "parentesis" 63 (figura 2). El circuito de reconocimiento de "parentesis" 63 tiene un par de conductores de entrada 175, 173, cada uno de los cuales se conecta al par de conductores de entrada en la línea 85. Las señales en cada uno de estos conductores 173 y 175 se alimentan a la entrada de la puerta O exclusiva 177 y, además, a una puerta Y 179 por la línea 193 y una puerta Y 181 por la línea 195. La señal de la puerta O exclusiva 177 en la línea 191 se alimenta como la otra entrada a las puertas Y respectivas. La salida 89 de la puerta Y 179 genera una señal de cómputo ascendente +1, mientras que la puerta Y 181 en la línea de salida 91 genera una señal de cómputo descendente -1 al contador binario ascendente/descendente 65. El contador ascendente/descendente 65 suministra un cómputo binario en las líneas 187 al circuito lógico de interfase 61 de la fila de espera de entrada (figura 2) y recibe una señal de cronometración desde el circuito lógico de interfase 61 por la línea 199 del cable 93.

En la tabla A se ilustra las dos representación de bitios preferidas de los cuatro caracteres utilizados en todo el ordenador (figura 1). En la primera fila se ilustra el delimitador izquierdo del dato, llamado por conveniencia parentesis izquierdo que está representado por una señal alta en una primera línea y una señal baja en una segunda línea, recibiendose ambas señales prácticamente al mismo tiempo. En la segunda fila se ilustra un delimitador derecho del dato o paréntesis derecho que está representado por una señal en la primera línea y una señal baja

T A B L A A

(	-	10
)	-	01

1 - 1 1

0 - 0 0

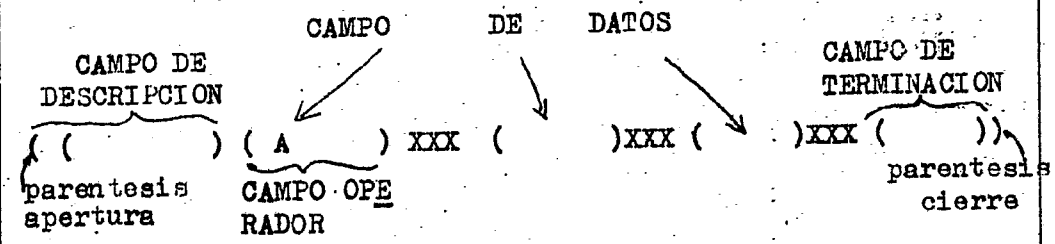
en la segunda línea, en oposición directa a la réprésentación del delimitador izquierdo del dato. Un caracter de 1 binario está representado por dos señales altas, como se ve en la tercera fila. Un caracter de cero binario está representado por dos señales bajas, segundo mostrado en la cuarta fila.

Refiriéndonos de nuevo a la figura 6, su funcionamiento para reconocer si las señales se transmiten a lo largo de la línea 85 representa un carácter delimitador del dato derecho o izquierdo, o un 1 binario o un 0 binario, se explica a continuación. Suponiendo, a título de ejemplo, que la señal binaria en el conductor 175 sea un 1, o señal alta, y la señal binaria en la línea 173 sea un 0, señal baja, la señal de la puerta O exclusiva 175 será un 1 binario y la señal en la línea 193 será un 1 binario, haciendo que la puerta Y 179 genere un nivel de señal alta en la línea 89. Este nivel de señal hace que el contador ascendente/descendente 65 cuente 1 en sentido ascendente. Suponiendo ahora que la señal binaria en la línea 175 sea un cero y la señal binaria en la línea 173 sea un 1, representando un caracter de paréntesis derecho, la salida de la puerta O exclusiva 177 será un 1 binario, haciendo que la salida de la puerta Y 181 en la línea 91 sea alta. El nivel de señal alta en la línea 91 hace que el contador binario ascendente/descendente 65 cuente uno en sentido descendente. El cómputo del contador binario ascendente/descendente 65, se alimenta al circuito lógico de interfase 61 de la fila de espera de entrada (figura 2). Siempre que ambas líneas de entrada 173 y 175 a la unidad de reconocimiento de "parentesis" 63 lleven señales elevadas, no se generará

señal de salida en las líneas 89 ni 91 porque la puerta O exclusiva 177 no genera una señal de activación en la línea 191. La misma situación existe cuando ambas líneas 173 y 175 son cero binario.

Refiriéndonos ahora a la tabla B, la disposición del campo o formato general del archivo de datos que es la unidad básica de una estructura de datos se ilustra en esta tabla. El primer campo de un archivo es un campo de descripción. Los campos inmediatamente siguientes son campos de datos. El último campo es un campo de terminación. Los paréntesis exteriores izquierdo y derecho, definen un archivo, Suponiendo que este archivo que puede considerarse una estructura de datos simple se transmita de izquierda a derecha. El primer campo que sigue al paréntesis de apertura es un campo de descripción que está delimitado en si por un par de parentesis. El campo siguiente que sigue al campo de descripción puede ser un campo operador como el ilustrado por el campo A, o un campo de dirección, o un campo operador.

T A B L A B



El dato en el campo de descripción describirá el tipo y orden de aparición de los diversos campos que siguen. Los espacios entre los campos de datos pueden llamarse, por conveniencia, "espacio vacío" que permite expandir los campos de datos si fuera necesario. Cuando estos campos se contraen crean más espacio vacio. Todo este espacio vacio puede utilizarse pa

ra permitir después que se expandan estos campos. El vehículo exacto por el que esto ocurre se describirá con mayor detalle más adelante.

El último campo de cada archivo es un campo de terminación que normalmente no llevará consigo dato. En otras palabras, es simplemente dos caracteres, un paréntesis izquierdo y un paréntesis derecho. El campo de terminación y el paréntesis de cierre de archivo son tres caracteres que representan el código de finalización para la estructura del dato o archivo. Este código entonces, según la convención de la tabla A es 100_{011}, transmitido en serie por caracteres o dos bitios en un instante paralelo de izquierda á derecha.

Este campo de terminación y el paréntesis de terminación de archivos se interpreta como un código de finalización de archivo por parte del circuito lógico de interfase 61 de la fila de espera de entrada (figura 2). Cuando aparece este código, la salida del contador 65 (figura 2) será cero, si no se han producido errores en los campos de los datos del archivo. Por ejemplo, el cómputo de salida del contador 65 para la estructura de archivo general de la tabla B, se producirá de esta manera, 121212121210. Así, una combinación de un cómputo cero desde el contador 65 y la aparición del código de finalización indica que la estructura de dato recibida no lleva consigo errores. Si, por ejemplo, hubiera un error en un caracter de paréntesis, el contador no experimentaría incremento ni decremento. Si hubiera un error en un carácter de dato, el contador de paréntesis incrementaria o experimentaría decremento de una forma incorrecta. En uno u otro caso, un contaje distinto a cero queda en el instante que aparece el código de terminación. Esto indicaría un error, haciendo que el circuito lógico de interfase de la figura 2, respondie

ra a la requisición de una retransmisión, según se ha descrito anteriormente.

La estructura de cada archivo, según se ilustra de un modo general en la tabla B, debe seguir ciertas reglas sintácticas. Estas reglas son:

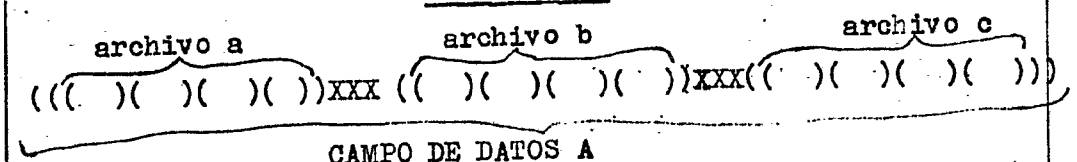
(1) No pueden aparecer caracter 1 o 0 entre parentesis iguales encarados. Por ejemplo, no puede haber caracteres entre el parentesis de apertura de archivo, y el parentesis de apertura de campo del campo de descripción.

(2) El primer campo de un archivo debe ser el campo de descripción

(3) El ultimo campo del archivo es siempre el campo de finalización. El nuestro ejemplo, este campo no lleva dato consigo.

Un campo de dato como es el campo de dato A de la tabla B puede estar compuesto en si por una pluralidad de campos o aún una pluralidad de archivos. Por ejemplo, la tabla C representa el campo A consistiendo en tres archivos auxiliares o subarchivos a, b, y c. El paréntesis de apertura de campo y el parentesis de cierre de campo definen el campo

T A B L A C



del dato A. Dentro de estos paréntesis puede aparecer una pluralidad de lo que se llamarán "archivos vectores". Los archivos a, b, y c, respectivamente, ilustran archivos vectores. Estos archivos, lógicamente, deben seguir las reglas sintácticas generales descritas para el archivo general de la tabla B. O sea, cada archivo lleva dentro de sí un campo de descrip

ción, campos de datos y campos de terminación. Como puede ocurrir dentro de un archivo, los espacios entre archivos vectores dentro de un campo XXX, pueden permitir la expansión de los archivos vectores dentro de dicho campo, si asi se desea.

Esta estructura de campos encajada dentro de archivos y archivos vectores dentro de campos se podrá comprender con más facilidad si se considera en términos de una estructura en árbol que tiene nodos que representan programas u operadores. A titulo de ejemplo, supongamos que la operación definida a continuación debe realizarse sobre una pluralidad de literales representados por las letras mayúsculas del alfabeto.

$$\left\{\left[(A+B-(C+D)\right]+\left[(F+G)-J\right]\right\}-\left\{\left(\left[(K-L)+(M-N)\right]+\left[O-Q\right]\right)-\right\}R-X$$

Esta combinación arítmetica de 14 literales diferentes puede estar representada por la estructura en árbol, ilustrada en la figura 7.

La estructura en árbol de la figura 7, recibe como entradas, a nivel de hoja 225, los literales, u otros operandos, sobre los que tienen que actuar el programa descrito por los diversos nodos 227, etc, del árbol. Así, por ejemplo los literales A y B se suministran al operador del programa de suma en el nodo 227; los literales C y D se suministran al operador del programa de suma en el nodo 229. Los resultados de ambas operaciones se alimentan a un operador de programa de resta en el nodo 231. Mientras esto ocurre, los literales F y G pueden suministrarse a otro operador de programa de suma en el nodo 235, alimentándose el resultado de dicha suma a un operador de programa de resta en el nodo 235,

alimentandose el resultado de dicha suma a un operador de programa de resta en el nodo 237, junto con otro literal J. Quizás al mismo tiempo que ocurre en estas operaciones anteriores los literales K y L se suministran a un operador de programa de resta en el nodo 239, suministrándose los literales N y M a otro operador de programa de resta en el nodo 241, y suministrándose los literales O y Q a otro operador de programa de resta en el nodo 247. El resultado de la operación en el nodo 239 y el resultado de la operación en el nodo 241 se suministran a un operador de suma en el nodo 243.

El resultado del nodo operador de resta 231 y el resultado del nodo de operador de resta 237 se suministran a otro nodo operador de suma 233. El resultado del nodo operador de suma 243 y el nodo operador de resta 247 se suministran a otro nodo operador de suma 245. El resultado del nodo operador de suma 245 se suministra al nodo operador de resta 249 que también se suministra a otro literal R. Los resultados del nodo operador de resta 249 y el nodo operador de resta 249 y el nodo operador de suma 233 se suministran a otro nodo operador de resta 251. El resultado de éste nodo se suministra a la operación envio a X 253.

Según resultará evidente por esta descripción de la estructura en árbol, el proceso de operandos en una estructura en árbol fácilita el proceso de operandos de una manera simultánea. O sea, las operaciones que tienen lugar en el mismo nivel que los nodos 227, 229, 235, 239, 241 y 247 pueden suceder de una forma prácticamente simultanea si se tienen disponibles los operadores apropiados. Los mismo ocurre con todas las operaciones en otro nivel, o segundo nivel, por ejemplo los nodos 231, 237 y 243, si el resultado de ope

raciones previas aparece disponible simultáneamente.

El ejemplo de la figura 4, con el fin de simplificar la descripción y facilitar la compresión, solamente considera operaciones diádicas como la suma y la resta. No obstante, se comprenderá que este tipo de flujo de proceso estructurado en árbol servirá para operaciones monadicas y diádicas con igual facilidad. Se comprenderá que para aprovechar las ventajas que supone el proceso simultáneo se debe utilizar un sistema de elaboración de datos.

Para ilustrar la forma en que las estructuras de datos de archivo agrupada de las tablas B y C ejecutan conceptos de proceso o elaboración estructural en árbol, consideraremos las operaciones diádicas simples siguientes sobre cuatro literales: (A+B - (C+D).

Estas operaciones se ilustran en forma estructurada en árbol en la figura 1. Los literales A,B.C y D a nivel de hoja 255, 257, 259, 261, se suministran al primer nivel de nodos operadores, los nodos de suma 263 y 265. Los resultados de éste nivel de nodos se alimentan al nivel siguiente o nodo de resta 267. El resultado 269 de éste nodo puede enviarse a otro nodo, u operador de programa, o a un destino físico. Cada nodo de la estructura en árbol tabla D, puede considerarse como un archivo, o serie de datos. Por lo tanto, observando estos dos niveles de operadores de nodo, el archivo que describiría el nodo de resta 267 se ilustra como archivo de nodo de resta. Este archivo está delimitado por parentesis derecho e izquierdo, y tiene un primer campo que es un campo de descripción, que describe la naturaleza y secuencia del archivo. En este caso P representa programa, indicando que este archivo es un archivo operador de programa. Como es-

te archivo es un archivo operador, el siguiente campo que sigue al campo de descripción será un campo que contiene el código operador OP. En nuestro ejemplo, el código operador describe una operación de resta. Como la operación es diádica, los campos que siguen al campo operador describen los dos operados que se han restar. Estos dos operandos son los resultados de los nodos de suma 263 y 265.

T A B L A D

archivo vector d — archivo vector e

(P)(OP)XXXX(V)(OP)(A)(B)(R)( ) XXXX(V)(OP)XX(C)(D)(R)( ) XXXXX(DA)XX( )

NODO DE RESTA

Como los operantes son resultados de otras operaciones, los campos operados son archivos vectores. Por lo tanto, los operandos se describen por los archivos vectores d y e. El campo que sigue a los campos de operandos es un campo de localización de destino DA que indica el destino al que debe enviarse el resultado de la operación de resta. El último campo del archivo de resta es el campo de finalización. Entre los campos dentro de un archivo puede aparecer en cualquier lugar un espacio vacío. Por ejemplo, dentro del archivo de programa de resta se ilustra el espacio vacio en varias posiciones X. Se recordará que como los campos de operadores del archivo de programa de resta son archivos vectores, también puede aparecer un espacio vacío entre los campos dentro de estos archivos.

Consideremos ahora los dos archivos vectores dentro del archivo del programa de resta, el archivo vector de suma d y el archivo vector de suma e. Estos archivos se estructuran de nuevo de acuerdo con las reglas sintácticas descritas

anteriormente. Existen paréntesis de delimitación de archivo izquierdo y derecho. Dentro de estos paréntesis, el primer campo es un campo de descripción que, en este caso describe el archivo como un archivo vector, reservando por lo tanto el campo inmediatamente siguiente para el código operador. Para nuestro ejemplo, se describe una operación de suma. Los campos que siguen al campo OP serán los campos de operadores que, en nuestro ejemplo son literales. Además de los campos de operandos, los archivos vectores diádicos como son los archivos d y e, dentro de un archivo mayor, como es el archivo de programa de resta, contienen campos resultantes, indicados por R en la Tabla D. Estos campos de resultantes (R) almacenan el resultado de la operación diádica descrita por dicho archivo vector si dicho resultado no se puede utilizar en el momento en que se genera.

Para facilitar la compresión, se describirá el funcionamiento general del ordenador 11 de la figura 1 con relación al flujo de programa simple ilustrado en la Tabla D, que solo utiliza operadores diádicos. Para facilitar además la explicación y compresión, supondremos que las estructuras de datos del programa o archivos del programa son diádicos y se reciben en la fila de espera de entrada 21 (figura 1). No obstante, se comprenderá que lo inverso tiene igual aplicación y que pueden almacenarse archivos de operandos en la memoria del ordenador 25 y se pueden suministrar archivos de programa al ordenador 11 por medio de la fila de espera de entrada 21.

Para realizar el flujo de funciones de la Tabla D, la memoria o almacenamiento del ordenador contendrá un archivo del programa según se ilustra en la Tabla E, bajo el encabezamiento "almacenamiento". El contenido inicial de este archivo,

antes de que el ordenador reciba cualquier archivo de operador se ilustra en la posición "1". El primer campo a de este archivo es un campo de descripción, que identifica el archivo como un archivo de programa. El primer campo b, que sigue a este campo de descripción es un campo que describe la operación que se ha de realizar. Para nuestro ejemplo, esta es una operación de resta. El campo siguiente que sigue al campo operador b es un campo operando delimitado por un paréntesis izquierdo d y un paréntesis derecho i. Este campo operando es un archivo vector que presenta una operación diádica. A este campo operandos sigue un segundo campo operando que es también un archivo vector. El campo que precede inmediatamente al campo de finalización es un campo de dirección de destino n. Se recordará que puede aparecer espacio entre los diversos campos del archivo de programa de resta por lo que los espacios X sirven para la expansión de los campos de operadores.

Consideremos ahora el primer campo de operandos, que es un archivo vector. En este caso particular, se define una operación de suma. Los campos de operandos e y f de este campo particular, como define una operación diádica, siguen al campo que describe al operador. Además, este archivo vector contiene un campo de resultado g en lugar de un campo de dirección de destino. Los campos de operandos e, f y el campo de resultado g se encuentran todos en estado contraído dejando una cantidad considerable de espacios vacíos X entre sí. En otras palabras, los campos están definidos simplemente por un paréntesis izquierdo seguido de un paréntesis derecho sin caracteres entre medias. Estos campos de operandos permanecen contraídos, según se describirá con más detalle más adelante, hasta que se almacenan en los mismos operandos.

T A B L A E

$$\overset{a}{\}}\ \overset{b}{\}}\qquad \overset{d}{\}}\qquad \overset{e}{\}}\qquad \overset{f}{\}}\qquad \overset{g}{\}}\qquad \overset{h}{\}}$$

((P)(RES)XXXXX ((V)(SUM)XXX ( )XXX ( )XXXXXXXXXX ( )XXX( ))

$$\overset{a_1}{\}}\qquad \overset{b_1}{\}}\ \overset{c_1}{\}}$$

((P)(RES)XXXXX ((V)(SUM)XXX(A)X ( )XXXXXXXXXX ( )XXX ( ))

((P)(RES)XXXXX ((V)(SUM)XXX ( )X ( )XXXXXXXXXX ( )XXX( ))

$$\overset{a_3}{\}}\qquad \overset{b_3}{\}}\qquad \overset{c_3}{\}}$$

((P)(RES)XXXXX ((V)(SUM)XXX ( )XXX( )XXXXXXXXX(RA+B)X( ))

$$\overset{a_4}{\}}$$

((P)(RES)XXXXX ((V)(SUM)XXX ( )XXX ( )XXXXXXXXX( )XXX( ))

MEMORIA DE DESTINO

$$\overset{a_9}{\}}\qquad \overset{b_9}{\}}\ \overset{c_9}{\}}\ \overset{d_9}{\}}\qquad \overset{e_9}{\}}\qquad \overset{f_9}{\}}\qquad \overset{h_9}{\}}$$

I. (DA) = ((L)(A)(OL) x ( )XXXX( ))

$$\overset{j_9}{\}}\qquad \overset{k_9}{\}}$$

(A) = (UNIDAD PERIFERICA)

$$\underset{j_9}{\{}\qquad \underset{l_9}{\{}\qquad \underset{m_9}{\}}$$

(A) = ((COMPUTADOR)( ))

El segundo campo de operadores para el archivo de programa de resta es también un archivo vector estructurado de la misma manera que se ha descrito para el primer campo de operando. Hay un par de campos de operando j y k, un campo de resultador l y un campo de finalización m. Cuando estos campos están vacíos, se encuentran en estado contraído dejando una cantidad considerable de espacios vacíos X, entre sí.

Lo anterior describe la estructura contemplada de un archivos de programa dentro de una memoria o almacenamiento de ordenador que permanece estática en almacenamiento hasta que una estructura o archivo de datos operandos llega a la fila de espera de entrada que localiza este archivo de programa particular. La estructura de este archivo de programa proporciona un mecanismo recursivo que acelera la ejecución de algoritmos.

Otra variante de estructura de dato para realizar la función de la Tabla E sería aquella que utiliza tres archivos de programa, en lugar de un archivo de programa, conteniendo dos archivos vectores, según se ilustra. Así, los dos archivos vectores de suma y el archivo de programa de resta representan tres archivos de programa independientes. El campo resultante (R) de cada archivo vector se reemplazaría por un campo de dirección o localización de destino (DA). El campo de dirección o localización de destino en ambos archivos de programa de suma localizaría el archivo de programa de resta, de la forma que se describirá más adelante. La utilización de este tipo de estructura de dato exige que el resultado de cada operación se dirija desde el ordenador de nuevo a su entrada para obtener el nodo operador siguiente. Por el contrario, la estructura de archivo de programa ilustrada elimina la necesidad de enviar el resultado de una operación de archivo vector fuera del

elaborador y de nuevo a su entrada para elaboración adicional.

Para continuar con la estructura de dato ilustrada, consideremos ahora los archivos de datos que llegan a la fila de espera de entrada. Supongamos que llega en un archivo de dato es el operador A. El archivo que contiene este operador se ilustra en la Tabla F como estructura de archivo 1 bajo el encabezamiento "fila de espera de entrada". El primer campo de este archivo de datos es un campo de descripción a_5 que indica que este archivo particular es un archivo operando que contiene

T A B L A F

FILA DE ESPERA DE ENTRADA

	a_5	b_5 c_5	d_5	e_5
1	((L)	(SA)(OL) X	(A) XXXX	( ))
	a_6	b_6	c_6	
2	((L)	(SA)(OL) X	(D) XXXX	( ))
	a_7	b_7	c_7	
3	((L)	(SA)(OL) X	(B) XXXX	( ))
	a_8	b_8	c_8	
4	((L)	(SA)(OL) X	(C) XXXX	( ))

un literal. Este campo de descripción es analizado por el analizador de campo 146 (figura 4) del control 23, que en respuesta al mismo establece los trayectos apropiados a la memoria del ordenador 25 para el siguiente campo b_5 que es un campo de dirección de almacenamiento que localiza el archivo vector particular al que pertenece el literal A. La dirección de almacenamiento b_5 localizará el lugar en la memoria del ordenador que comienza con el paréntesis izquierdo d del archivo vector de suma dentro del archivo de programa de resta b. El campo siguiente inmediatamente después del campo de dirección b_5 es un

campo de localización de operando c_5 que indica si el campo de operador d_5 que sigue pertenece al campo de operador de izquierda o derecha e o f, respectivamente, de dicho archivo vector particular. El archivo de operando que se recibe en la fila de espera de entrada tiene un campo de finalización e_5 y puede tener un espacio vacío XXXX entre los campos de dicho archivo.

El control 23 por medio de su ROM analizadora de campo 146 y su librería de su rutinas consistentes en la pluralidad de memorias ROM 154, 156, 158 interroga al archivo vector de suma, después que ha sido localizado por el archivo de operandos en la fila de espera de entrada, para determinar si el operando B ha llegado anteriormente y está almacenado dentro de su campo f. En este caso, no se ha almacenado, según indica al control los campos de operandos vacíos e, f, el control almacena el operando A en el campo apropiado e. Cuando el operando A se escribe en la memoria, caracter por caracter, el archivo de operando e se expande para acomodarse a su tamaño exacto. La forma específica de como el operando se escribe realmente en la memoria queda dentro de los conocimientos del experto en la materia por lo que no se explicará en la presente memoria.

Por lo tanto, como resultado del archivo de literales ilustrado en la posición 1, que llega a la fila de espera de entrada, el archivo de programa de resta en la memoria del ordenador detendrá un literal A almacenado dentro del campo de operando apropiado b_1 del archivo vector que ha sido localizado y comienza con el paréntesis izquierdo a_1, según se ilustra en la posición 2 bajo el encabezamiento "almacenamiento" de la Tabla E. Como el literal A está ahora almacenado dentro de su campo de operando apropiado, dicho campo se ha expandido y el espacio vacío X entre este campo de operador y su campo de ope

rador compañero puede utilizarse completamente o disminuirse notablemente. Supongamos ahora que el archivo de operandos siguiente que entre en la fila de espera de entrada 21 del ordenador 11 (figura 1) contiene el operador D en su campo de operando c_6, según se ilustra en la posición 2 bajo el encabezamiento "fila de espera de entrada". Además del campo de descripción que indica a la unidad de control los campos que han de seguir, un campo de localización de almacenamiento a_6 y un campo de localización de operando b_6 están presentes en este archivo de operandos. El archivo de literales en la posición 2 de la fila de espera de entrada tiene un campo de localización de almacenamiento a_6 que localiza el archivo vector de suma dentro del archivo del programa de resta en el paréntesis inicial d_1 (posición 2 después del encabezamiento de "almacenamiento"). Una vez que se ha localizado este archivo vector, la unidad de control, al observar el campo de operador del archivo vector hara que se active el microprograma de suma apropiado de la libreria de microprograma compuesta por los ROM 154, 156 y 158 (figura 4). Si este microprograma detecta que todos los operandos que son necesarios para realizar la operación no están presentes, bien en la fila de espera de entrada o el almacenamiento o memoria del ordenador, se activa otro microprograma para almacenar el literal D en el campo de operando c_6 de la fila de espera de entrada en el campo de operando apropiado a_2 del archivo vector de suma, según determine el código de lugar de operando en el campo b_6 del archivo de literales en la fila de espera de entrada. Como resultado de el segundo archivo de literales, la estructura del dato en almacenamiento aparecerá según se ilustra en la posición 3 bajo el encabezamiento de "almacenamiento". O sea, se almacena un literal A en su campo de ope-

rando apropiado en el primer archivo vector de suma y se almacena un literal D en su campo de operando apropiado en el segundo archivo vector de suma.

Supongamos ahora que el tercer archivo de operando que entra en la fila de espera de entrada lleva un operando B en el campo de operando c_7 que se ha de combinar con el operando A. El controlador reconoce, debido al campo de descripción L, que este es un archivo de literales y, por lo tanto, el campo siguiente a_7 es una dirección o localización de almacenamiento que localiza el primer archivo vector que contiene el operando A. El controlador procede a tomar lectura de este archivo vector localizado, y su ROM analizadora de campo 146 (figura 4) determina por el campo de descripción "V" que es un archivo vector que contiene un programa. El campo que debe seguir este campo de descripción es entonces un campo de código operador. En respuesta al campo operador, el analizador de campo hace entrar en acción a un microprograma apropiado de la libreria de microprogramas ROM 154, 156 o 158 (figura 4) y además hace que se tome lectura de la memoria del literal A para localizar la ROM apropiada 125 en la unidad lógica vectorial (Figura 3), al par que hace que se tome la lectura del literal B de la fila de espera de entrada para localizar la misma ROM 125 en la unidad lógica vectorial.

Se recordará que la unidad lógica vectorial es una unidad arítmetica serial que actúa sobre dos caracteres al tiempo, un caracter de cada uno de los dos campos de operandos. Cuando la unidad lógica vectorial ha completado su función de sumar los operandos A y B entre sí, el microprograma determina si el campo resultante en el segundo archivo vector está lleno. Como en este caso está vacío, almacenará el resul-

tado de la adición de literales A y B en el campo resultante apropiado en el primer archivo vector. Como resultado de que aparezca el tercer archivo de operador en la fila de espera de entrada, el archivo de programa de resta en almacenamiento se estructurará según se indica en posición 4 bajo el encabezamiento "almacenamiento", o sea, los campos de operandos que ocupan los literales A y B, campos a_3 y b_3, respectivamente, está ahora vacios, puesto que se contrajeron al tomarse la lectura, y el campo resultante a_3, que contiene el resultado de la suma de A y B, está lleno. El literal D como operando del segundo archivo vector también está presente.

El único operando que falta, en este instante, es literal C. Supongamos ahora que un archivo de operando aparece conteniendo el operando C en el campo c_8. El control reconoce que este es un archivo de literal y cierra los trayectos apropiados de forma que el campo de localización de memoria a_8 pueda localizar el segundo archivo vector. El control tomará entonces lectura de este archivo vector, pondrá en condiciones a la unidad lógica vectorial para realizar la operación requerida por el campo del código operador y procederá a sumar C y D de la misma manera que se ha descrito para los operandos A y B. No obstante, al completarse esta operación como el campo de resultante a_4 del primer subarchivo de programa está lleno, además de almacenar el resultado de la suma de los literales C y D en el campo resultante d_4, se elige otro microprograma que ponga en condiciones a la unidad lógica vectorial de acuerdo con el campo de código de operador de resta en el archivo de programa de resta. Este microprograma hace que la unidad de control abastezca a la unidad lógica vectorial, de una manera serial por caracteres, la resultante de la suma A+B del campo

de resultantes a_4 en la memoria del ordenador cuando se suministra a la misma el resultado de C+D para que los dos resultados se resten.

Mientras se realiza esta operación, el campo de destinos n del archivo de programa de resta se suministra a la memoria de dirección de destino 157 de la fila de espera de salida 29 (figura 5). Este campo de dirección de destino a_9, según se ilustra en la Tabla E bajo el encabezamiento "fila de espera de salida" en la posición 1, es un archivo vector de destino que tiene como su primer campo un campo de descripción b_9, que, en nuestro ejemplo, identifica el archivo como un archivo de literal u operador; un campo de dirección c_9 que lo sigue; y un campo de lugar de operando d_9 que sigue al campo de dirección. Los campos de operandos, como es el campo e_9 pueden seguir al campo de lugar del operando. Como debe seguirse las sistaxis de una estructura de archivo, el archivo de dirección o localización de destino termina con un campo de finalización f_9. El campo de dirección de destino, como es un archivo de vector, puede tener también espacio vacio entre los campos, como es el espacio vacío XXXX, por ejemplo.

El campo de operando e_9, en este punto, no tiene nada almacenado en su interior y está en forma contraída. Cuando la unidad lógica vectorial obtiene el resultado de restar los literales C+D de los literales A+B, dicho resultado según se ilustra en la posición 1 bajo el encabezamiento "memoria de operando" de la Tabla E se envía a la memoria de operadores 155 de la fila de espera de salida (figura 5).

El control de salida 159 de la fila de espera de salida 29 (figura 5) transmite un mensaje en una forma que es esencialmente idéntica a la forma que se recibe en la fila de espe

ra de entrada según se ilustra en la Tabla E bajo el encabezamiento "mensaje transmitido". Como, en nuestro ejemplo, el resultado es un literal, el archivo transmitido es un archivo de operando, delimitado por un paréntesis derecho g_9 y un paréntesis izquierdo h_9. El primer campo es un campo de descripción i_9 que define el archivo como un archivo de operador. El segundo campo es un campo de dirección j_9. Este campo de dirección, según se ilustra en la Tabla E, puede ser un campo simple que contiene una designación de unidad periférica k_9 o, cuando se trata de un sistema de elaboradores múltiples, puede contener campos compuestos, como es el campo l_9, que definen una unidad elaboradora y un campo de dirección o localización de almacenamiento m_9, que define un área específica en el almacenamiento del elaborador localizado. El campo que sigue al campo de dirección es un campo de localización de operando n_9, si fuera necesario. El campo que sigue al campo de lugar del operando es el campo de resultado o_9. El archivo de operando que sale de la fila de espera de salida finaliza con un campo de finalización p_9 y un paréntesis derecho q_9.

En resumen, la descripción funcional anterior pone de manifiesto que el ordenador de la figura 1 ejecuta una operación solamente después de enlazarse dos estructuras de dato, de las cuales una es una estructura de programa y la otra es una estructura de operando. En el caso del ejemplo específico, la estructura de programa en forma de archivo de programa se almacena en la memoria del ordenador esperando la llegada de las estructuras de operandos o archivos de operandos que localizan los archivos de programa apropiados, haciendo que la unidad de control del ordenador ejecute el programa indicado. Esta operación activada por datos, por lo tanto, proporciona un

elaborador de datos digitales que tiene capacidades de emulación superiores y puede utilizarse como un bloque de construcción básico en un ordenador de procesadores múltiples, teniendo cada uno de los bloques de construcción sus funciones definidas por los archivos de programa almacenados dentro de sus áreas de almacenamiento respectivas. Como la llegada de archivos de operandos en la entrada de un elaborador de datos específica produce la activación del programa localizado cuando dicho ordenador se utiliza como bloque de construcción en un ordenador de elaboradores múltiples, no sería necesario un programa de control maestro o un estenso sistema de interrupciones que regularán la interacción de los procesadores dentro del ordenador de elaboradores múltiples, La descripción anterior, según se observará, pone de evidencia que el ordenador de la figura 1 tiene capacidades de emulación mejoradas porque el vocabulario de cuatro caracteres en el ordenador serial por caracter facilita una estructura de dato de longitud de campo variable. Estas estructuras de datos pueden comprobarse fácilmente para hallar errores mediante la utilización de circuitos lógicos simples en los trayectos de flujo de datos.

Como es lógico, se comprenderá que la descripción anterior se refiere solamente a una modalidad de preferencia del invento y que pueden realizarse numerosas modificaciones sin desviarse del espíritu y alcance del invento.

N O T A

Descrita suficientemente la naturaleza del invento así como la manera de realizarlo en la práctica, debe hacerse constar que las disposiciones anteriormente indicadas son sus-

ceptibles de modificaciones de detalle en cuanto no alteren su principio fundamental. También se hace constar que el invento corresponde a unas solicitudes de patentes presentadas en Norteamerica con fechas 28 de Febrero de 1.974 y 13 de septiembre de 1.974, bajo los números Ser. No. 446.912 y 505.869, acogiéndose por lo tanto a los beneficios que conceden los Convenios Internacionales en vigor, siendo lo que constituye la esencia del referido invento y por lo que se solicita Patente de Invención por 20 años en España sobre: PERFECCIONAMIENTOS EN DISPOSITIBOS PROCESADORES DE DATOS; caracterizándose por lo siguiente:

1ª.- Perfeccionamientos en dispositivos procesadores de datos binarios del tipo que tienen medios de almacenamiento y medios de circuito de entrada, caracterizados porque se dota al dispositivo de un mecanismo recursivo, que comprende estructuras de datos almacenados en los medios de almacenamiento, representando las estructuras de datos de programa organizados en un orden encajado jerarquíco de acuerdo con archivos de datos de programa compuestos por campos de operandos y un campo de resultantes asociados con un campo de descripción de programa particular, y estructuras de datos recibidos por los medios de circuito de entrada, representando las estructuras de datos operandos organizados en un orden encajado jerarquíco y produciendo las estructuras de datos la localización de ciertas estructuras de datos en los medios de almacenamiento.

2ª.- Perfeccionamientos, según la reivindicación 1, caracterizados porque un archivo de datos de programa tiene por lo menos un campo de operando y un campo de localización de destino asociados con un campo de descripción de programa, siendo dicho campo de operando un archivo vector compuesto por lo menos por un campo de operando, un campo de resultante, y un

campo de descripción vectorial.

3ª.- Perfeccionamientos, según la reivindicación 2, caracterizados porque los campos de operandos y el campo de resultante del archivo vector se contrae cuando se vacian hasta que se expanden por datos escritos en su interior.

4ª.- Perfeccionamientos, según la reivindicación 1, caracterizados porque los campos de operandos y los campos de resultantes se contraen cuando se vacian hasta que se expanden cuando se escriben datos en su interior.

5ª.- Perfeccionamientos, según las reivindicaciones anteriores caracterizados porque el formato de las estructuras de datos almacenados en los medios de almacenamiento, que representan datos de programa organizados en un orden encajado jerarquíco de acuerdo con archivos de datos de programa es el siguiente: ((P) (OP)xxx ((V) (OP)xx(A)xx(B)xx(R) ( ) xx(C)xx (DA) ()) donde las áreas delimitadas por paréntesis, dentro del paréntesis, son campos, representados P un campo de descripción de programa, representando OP un campo de operador representando A, B y C campos de operandos, representando V un campo de descripción vector, representando R un campo de resultante y representando DA un campo de localización de destino.

6ª.- Perfeccionamientos, según la reivindicación 5, caracterizados porque el campo de operando y el campo de resultante se contraen cuando se vacian hasta que se expanden cuando se escriben datos en su interior.

7ª.- Perfeccionamientos, según las reivindicaciones anteriores, caracterizados porque el formato de las estructuras de datos almacenados en los medios de almacenamiento, que representan datos de programa organizados en un orden encajado jerarquíco de acuerdo con un archivo de datos de programa es el

siguiente ((P) (OP)xxx((V) (OP)xx(A)xx()xx(R) ()xx(C)xx(DA) ()) donde todas las áreas delimitadas por paréntesis dentro de paréntesis son campos, representando P un campo de descripción de programa, representando OP un campo de operador, representando A y C campos de operandos, representando V un campo de descripción vector, representando R un campo de resultantes y representando DA un campo de localización de destino; y estructuras de datos recibidas por los medios de circuito de entrada que representan datos de operandos organizados en un orden encajado jerarquíco de acuerdo con archivos de operandos compuestos por los menos por un campo de operando y un campo de descripción de operando particular.

8ª.- Perfeccionamientos, según la reivindicación 7, caracterizados porque los campos operandos y el campo de resultante en los medios de almacenamiento se contraen cuando se vacian hasta que se expanden cuando se escriben datos en su interior.

9ª.- Perfeccionamientos en dispositivos procesadores de datos; tal y como queda sustancialmente descrito en la presente Memoria y en los adjuntos dibujos.

Esta Memoria, consta de cuarenta y seis hojas, escritas a máquina por una sola cara.

22 ABR. 1975

Madrid,

BURROUGHS CORPORATION

J. GOMEZ ACEBO Y MODET
p. p. Firmado: L. Gaeta Fernández

BURROUGHS CORPORATION.

EN 3 HOJAS Nº1

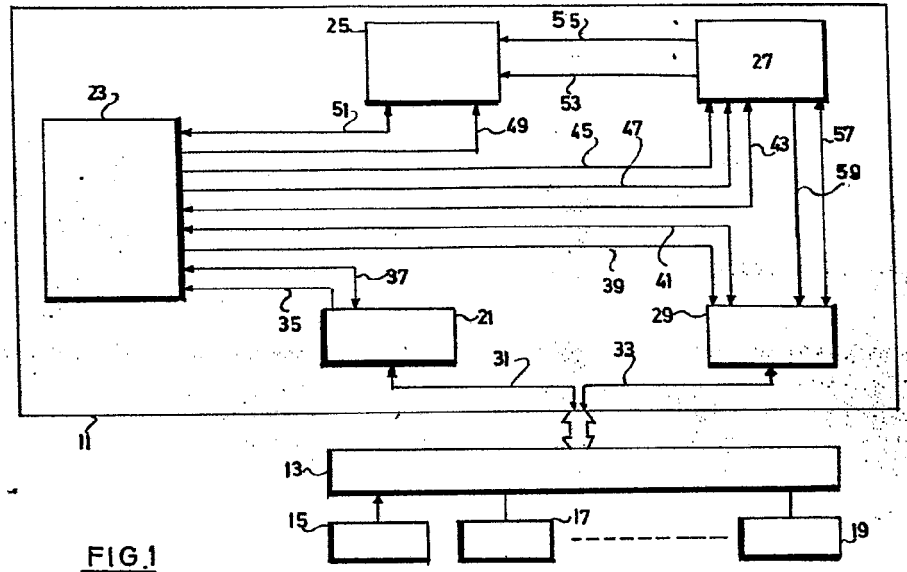


FIG.1

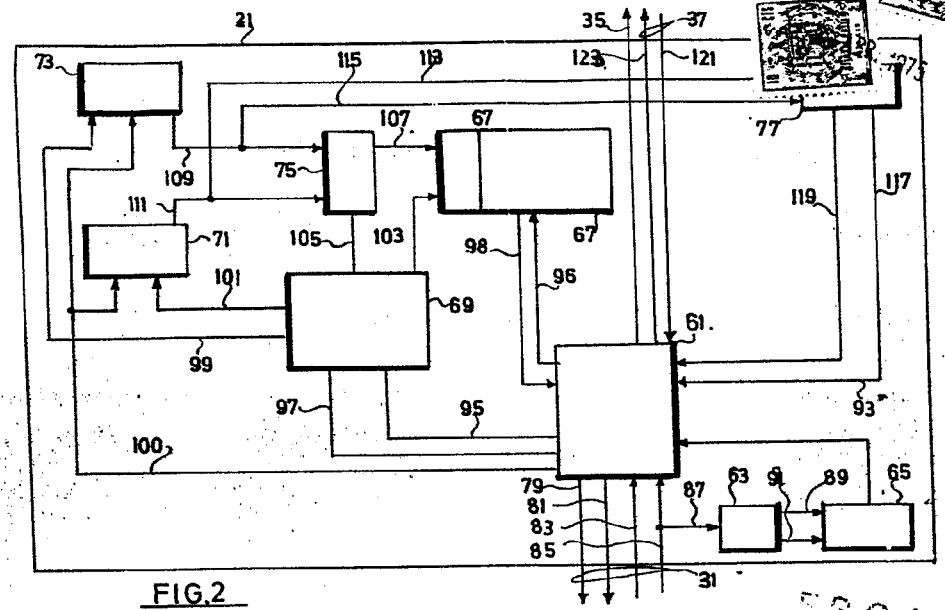


FIG.2

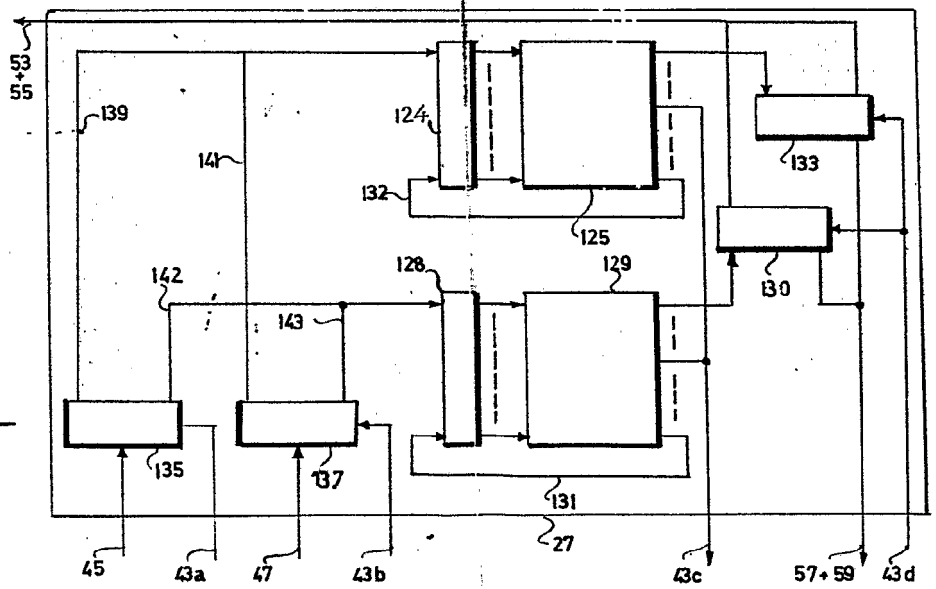


FIG.3

ESCALA VARIABLE.

Madrid, ABR. 1971

BURROUGHS CORPORATION.

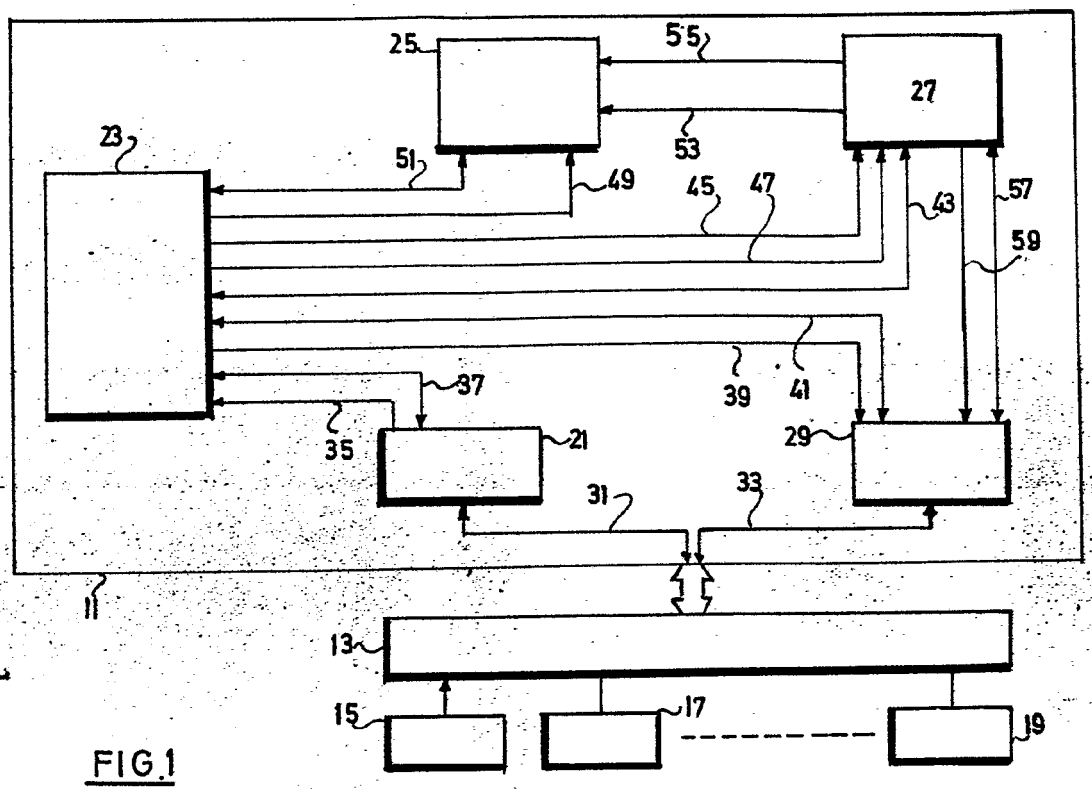


FIG.1

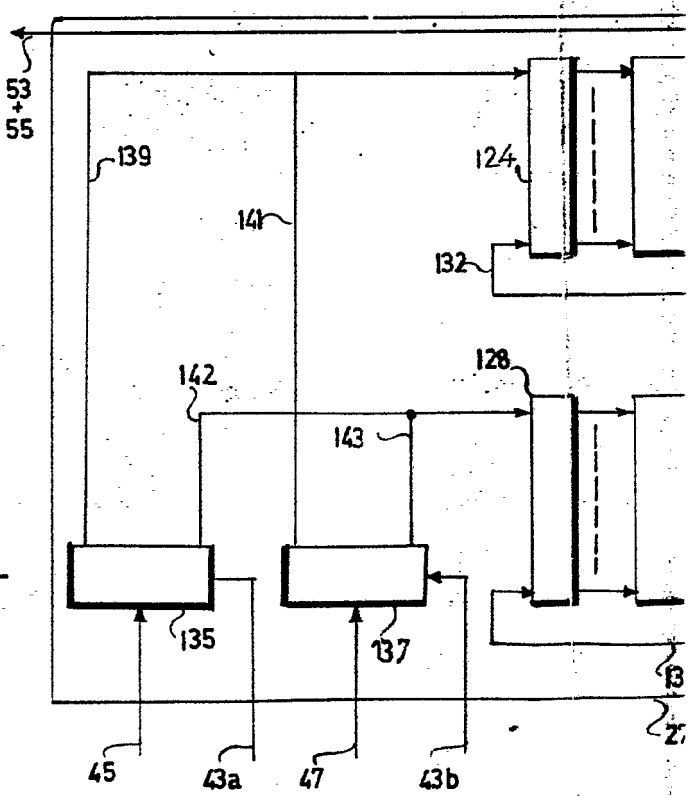


FIG.3

ESCALA VARIABLE.

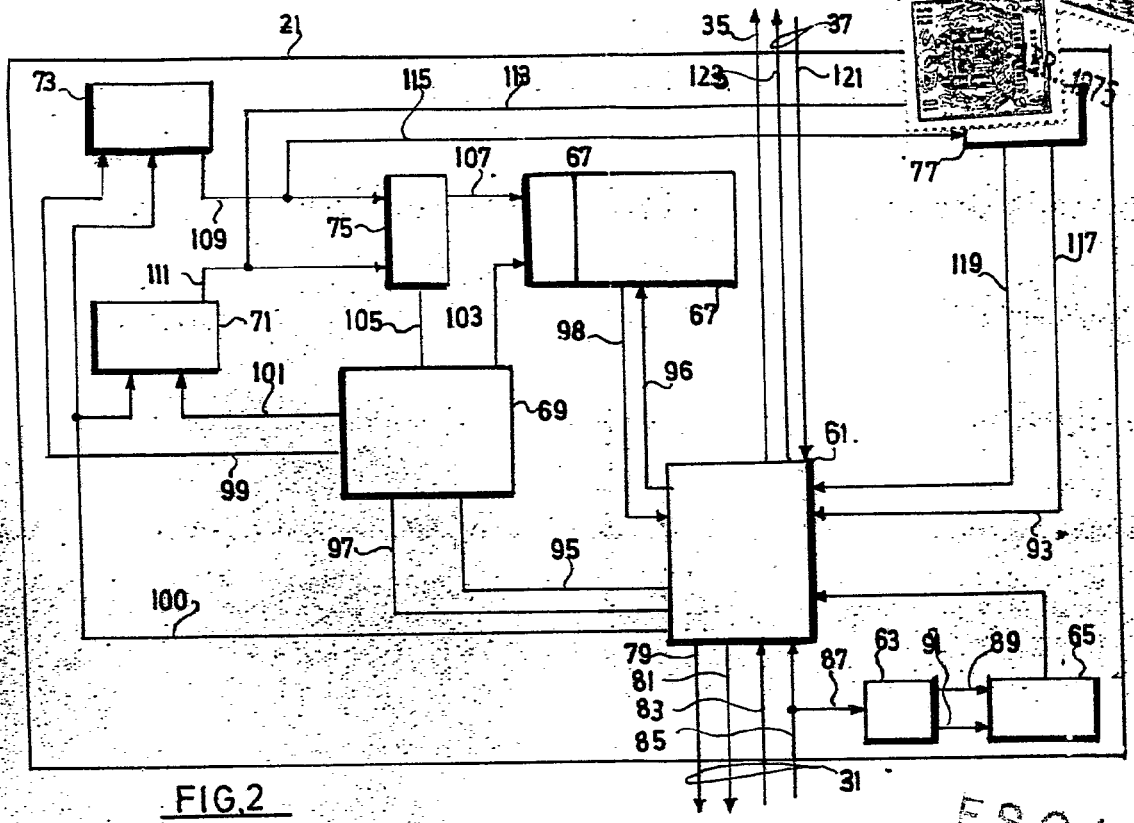
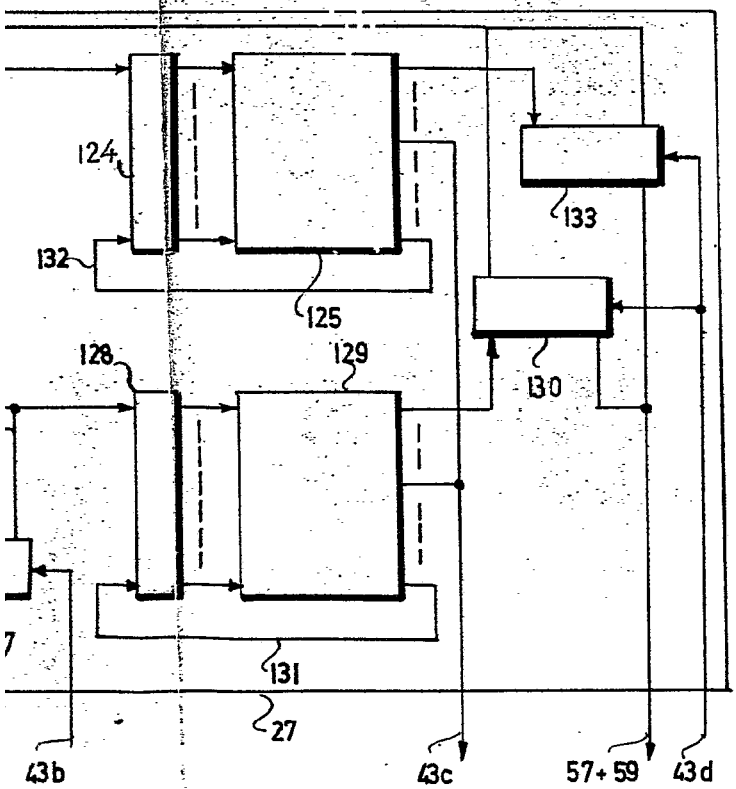


FIG.2



Madrid 22 ABR. 1975

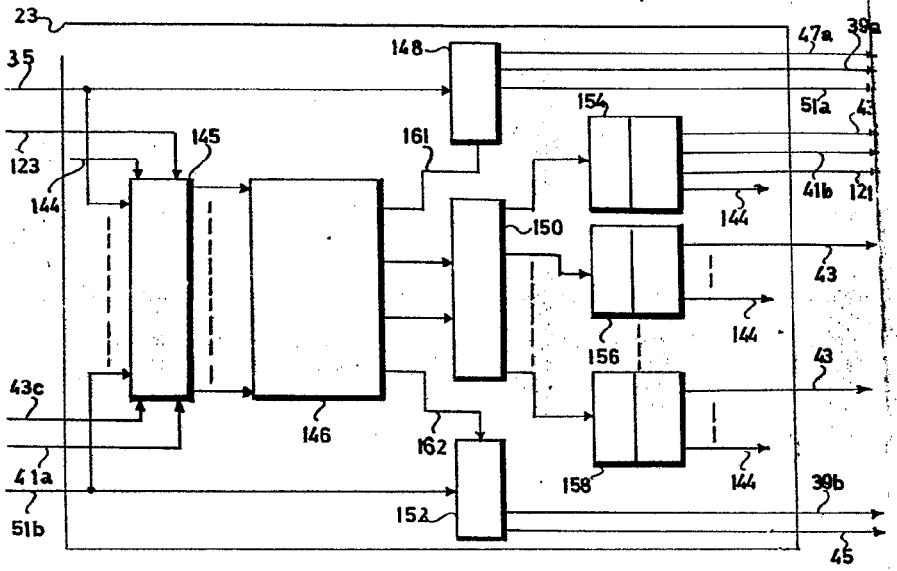


FIG.4

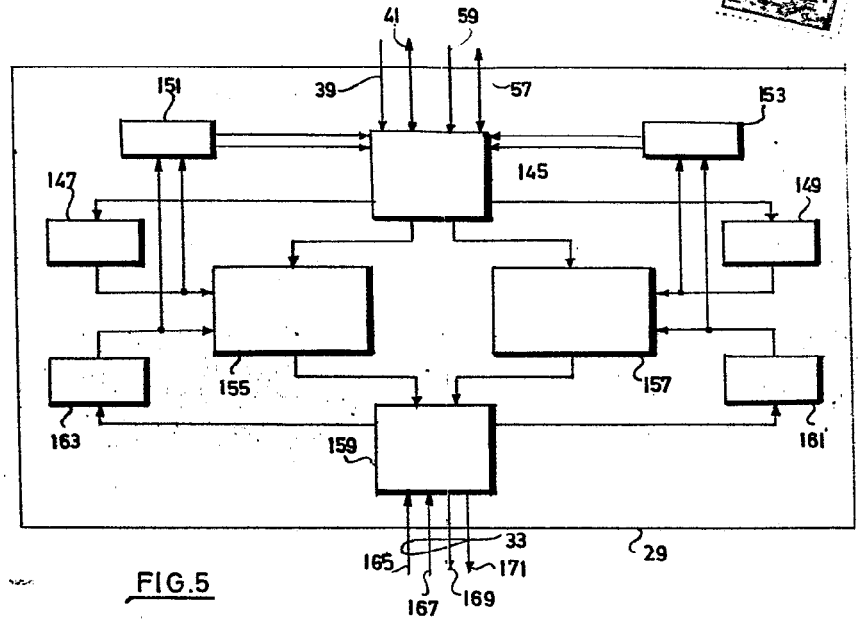


FIG.5

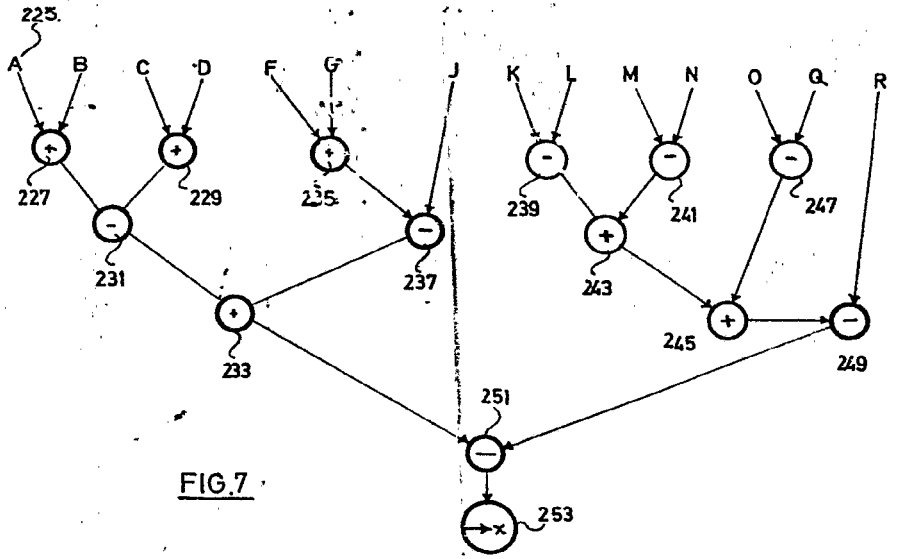


FIG.7

ESCALA VARIABLE.

22 ABR. 1975

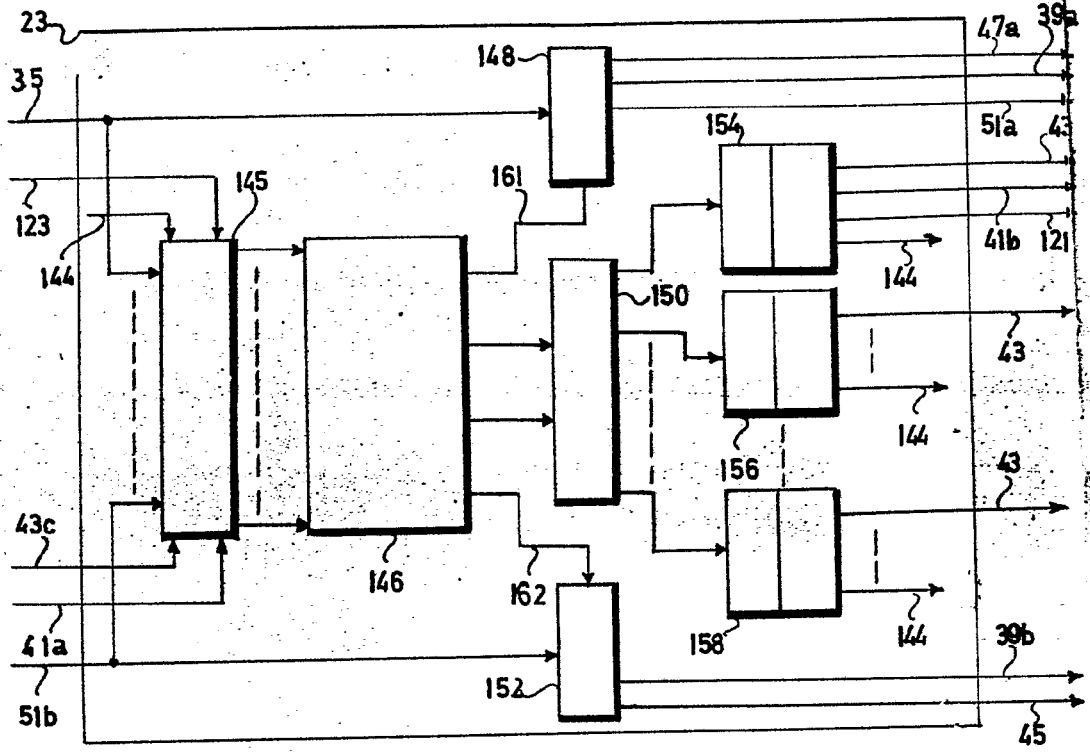

23
35
123
144
145
148
161
154
150
156
158
162
152
146
43c
41a
51b
47a
51a
41b
121
43
39a
39b
45

FIG.4

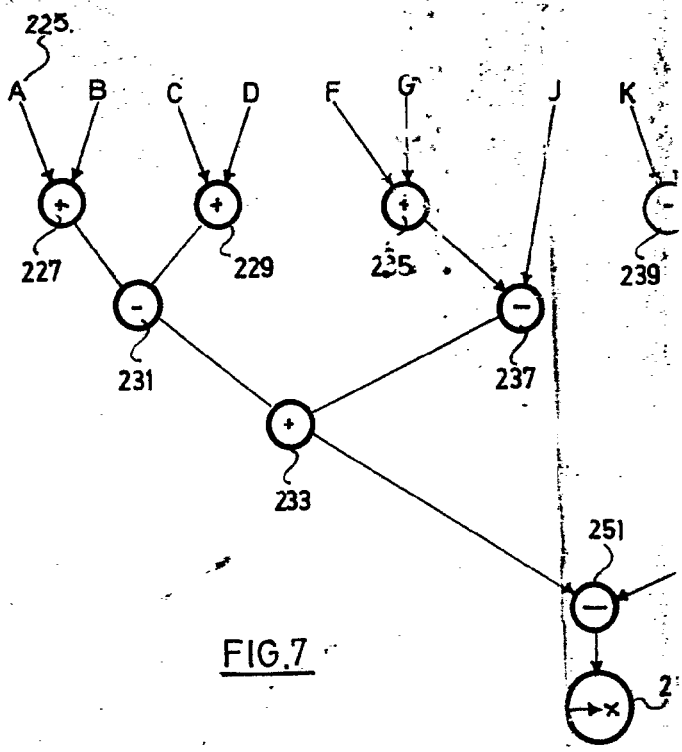

225
A
B
C
D
F
G
J
K
227
229
231
233
235
237
239
251

FIG.7

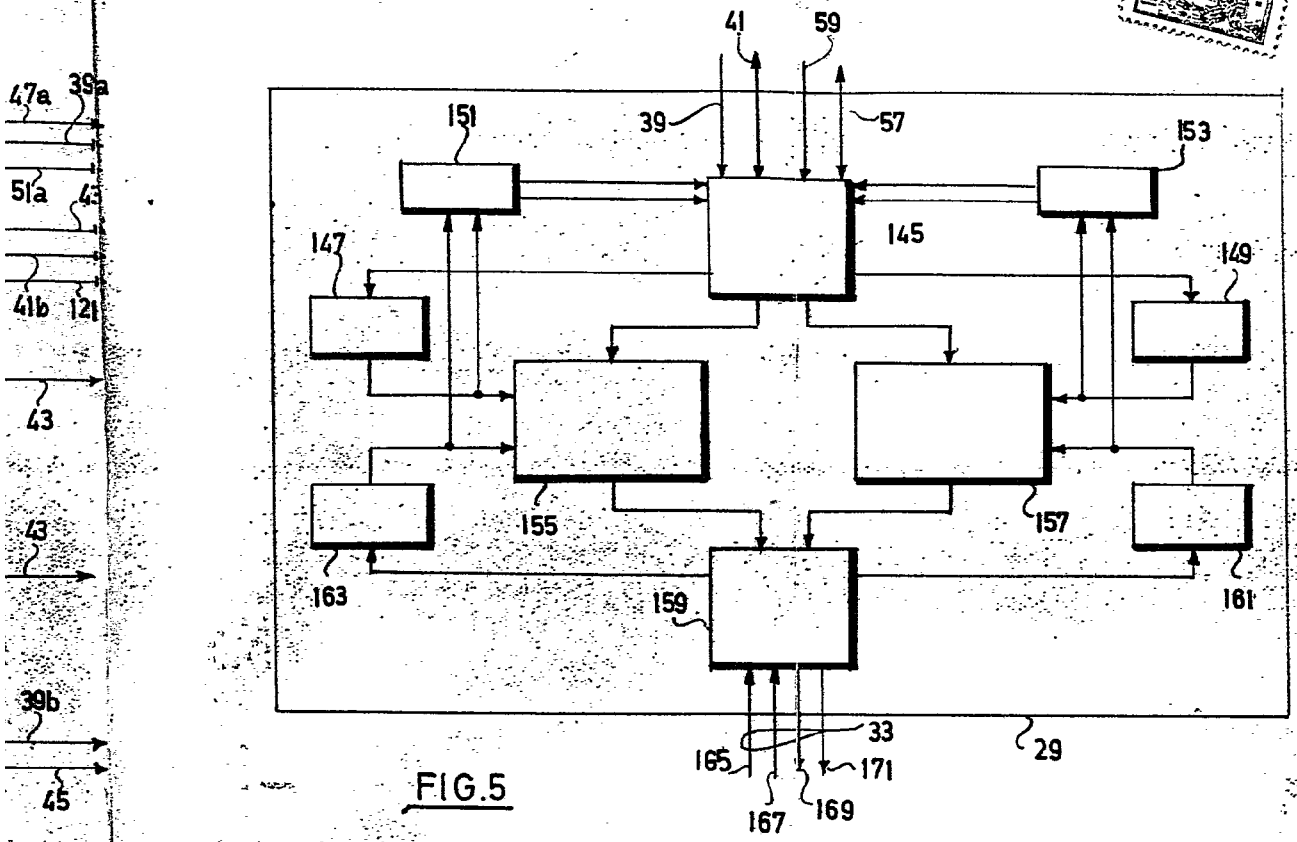
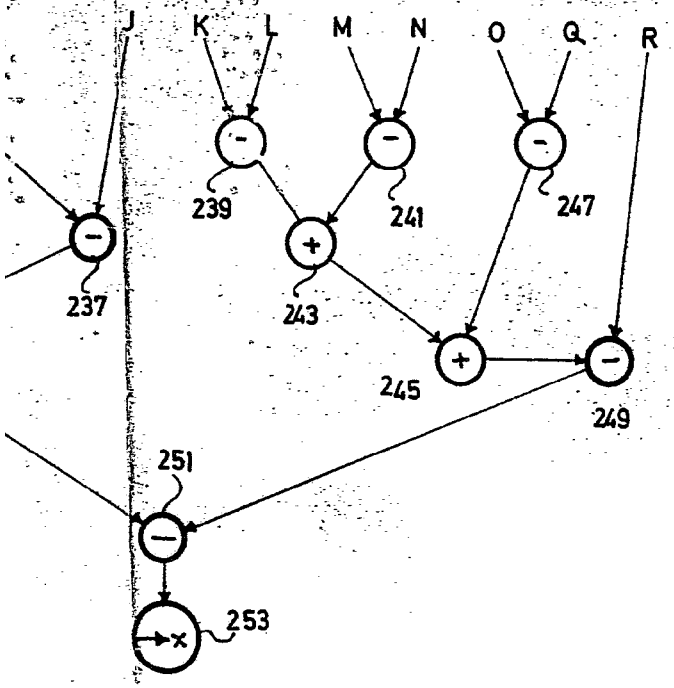


FIG.5



22 ABR. 1975

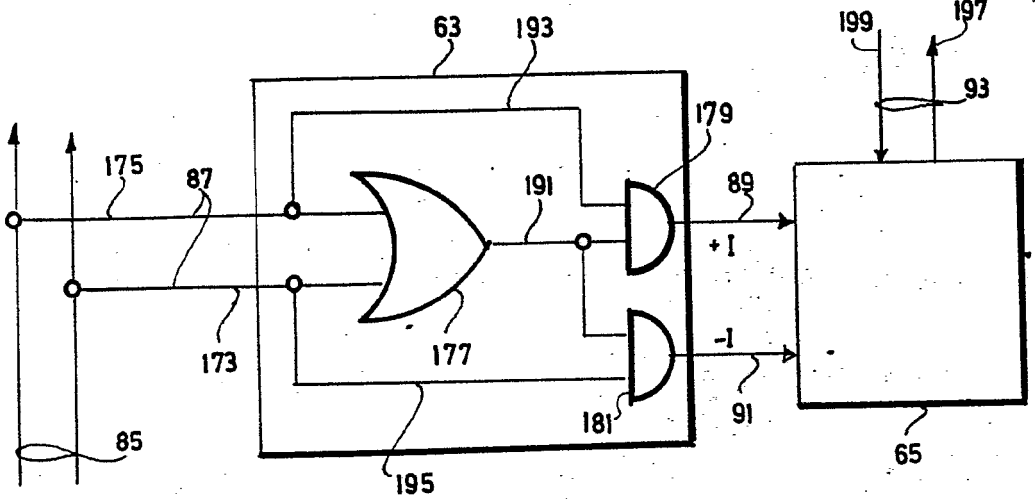


FIG.6

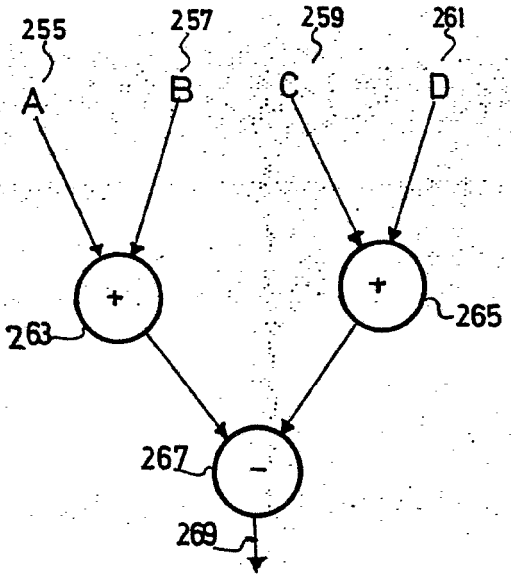


FIG.8

ESCALA VARIABLE.