CAPITULO 1

ARQUITECTURAS

1.1. - INTRODUCCIÓN

El tema de Arquitecturas aquí tratado no se enfoca desde el punto de vista de la ingeniería electrónica, sino desde el punto de vista funcional, ya que nuestra meta es llegar a mostrar qué componentes existen y cómo pueden ser amalgamados para cumplir una función específica más que analizar su estructura real.

Salvo casos expresamente identificados, el tema de Sistemas Operativos se referirá siempre a casos multitarea multiusuarios.

Realizaremos en esta primera parte todo un pantallazo sobre los temas de arquitecturas y sistemas operativos a efectos de homogeneizar conceptos y terminología.

1.2. - SISTEMAS OPERATIVOS

En esta primera parte trataremos de responder a preguntas del tipo : Qué es un Sistema Operativo, qué hace, y para qué sirve ?. De esta forma queremos mostrar dónde queremos llegar para ver posteriormente la forma en que lograremos esto en capítulos subsiguientes.

Las dos funciones primordiales de un S.O. son las siguientes :

1)- **La utilización compartida de recursos**

2)- **La constitución de una máquina virtual**

En el primer punto es importante saber que un sistema operativo debe lograr que se compartan los recursos de un computador entre un cierto número de usuarios que trabajan en forma simultánea (obviamente en sistemas multiusuario ó multitarea). Esto es importante ya que de esta forma se busca incrementar la disponibilidad del computador con respecto a esos usuarios y, al mismo tiempo, maximizar la utilización de recursos tales como el procesador central, la memoria y los periféricos de E/S.

La segunda función de un Sistema Operativo es la de transformar un cierto hardware en una máquina que sea fácil de usar. Esto es equivalente a colocar frente al usuario una **máquina virtual** cuyas características sean distintas y más fáciles de manejar que la máquina real subyacente.

Un ejemplo concreto y muy conocido en el cual la máquina virtual difiere de la real es por ejemplo en el campo de las E/S, ya que en este caso el hardware básico puede que sea extremadamente difícil de manejar por el usuario y que requiera de sofisticados programas. Nuestro Sistema Operativo deberá entonces evitar al usuario el problema de tener que comprender el funcionamiento de este hardware poniendo a su alcance una máquina virtual mucho más sencilla pero con las mismas posibilidades de E/S.

La naturaleza de una máquina virtual dependerá de la aplicación específica para la cual se la quiera utilizar. Evidentemente el diseño del Sistema Operativo de esta máquina estará fuertemente influenciado por el tipo de aplicación que se quiera dar a la máquina.

Existen máquinas de **propósito general** en las cuales no se puede identificar un tipo de aplicación concreta. Tales máquinas son muy criticadas en el sentido de que por tratar de ofrecer prestaciones en un espectro muy general de aplicaciones no pueden efectivamente responder eficientemente en ninguna de ellas.

1.2.1. - Funciones de un Sistema Operativo

Los sistema operativos deben llevar a cabo como mínimo las siguientes funciones :

- Secuenciar las tareas : llevar un cierto orden respecto de los diferentes trabajos que debe realizar.
- Interpretar un lenguaje de control : debe comprender los diferentes tipos de órdenes que se le imparten para poder ejecutar las tareas.
- Administrar errores : debe tomar las adecuadas acciones según los diferentes tipos de errores que se produzcan como así también el permitir la intervención externa de, por ejemplo, el operador.
- Administrar las interrupciones : debe interpretar y satisfacer todas las interrupciones que se puedan llegar a producir.
- Scheduling : administrar equitativamente el recurso procesador entre las diferentes tareas.
- Controlar los recursos existentes : debe llevar debida cuenta de todos los recursos como así también del estado en que se encuentran y las funciones que realizan en todo momento.
- Proteger : debe proteger la información de todos los usuarios entre sí, como así también debe asegurar la integridad de los datos.
- Debe permitir el procesamiento interactivo.
- Debe ser de fácil interacción para los usuarios.
- Debe llevar un control global de todos los recursos del sistema.

En virtud de esta enumeración podemos extraer de la lista anterior una serie de características que debe poseer un Sistema Operativo.

1.2.2. - **Características de un Sistema Operativo**

1.2.2.1. - **Concurrencia**

La concurrencia consiste en la existencia de varias actividades simultáneas o paralelas. Por ejemplo la superposición de las E/S con la ejecución del cómputo.

La concurrencia lleva asociado los siguientes problemas :

- Conmutar de una tarea a otra.
- Proteger una determinada actividad de los efectos de otra
- Sincronizar las tareas que sean mutuamente dependientes.

Definiremos el procesamiento en paralelo como :

Definición *: El procesamiento paralelo es una eficiente forma de procesar la información que pone énfasis en la explotación de eventos concurrentes en el proceso de cómputo. La concurrencia implica* ***paralelismo****,* ***simultaneidad*** *y* ***pipelining****.*

Los *eventos paralelos* pueden ocurrir en diferentes recursos en el mismo intervalo de tiempo, la multiprogramación es un ejemplo clásico de paralelismo ya que dado un intervalo de tiempo la CPU, vista desde la óptica del usuario, simula que todos los programas son ejecutados al mismo tiempo.

Los *eventos simultáneos* pueden ocurrir en el mismo instante del tiempo, por ejemplo un sistema que cuente con dos CPU's estaría ejecutando simultáneamente instrucciones en cada una de ellas.

Los *eventos pipeline* pueden ocurrir en intervalos de tiempo superpuestos, por ejemplo si se tuviera una unidad hardware interna a la CPU que interpretase el código de operación de una instrucción y otra unidad que cargase de memoria la próxima instrucción a ejecutar, mientras se decodifica el código de la instrucción actual se estaría cargando la próxima instrucción en la CPU, luego se superponen en el tiempo ambas funciones.

Estos eventos concurrentes se alcanzan en un sistema de computación con varios niveles de procesamiento. El procesamiento paralelo necesita de la ejecución concurrente de muchos programas en el computador. Esto es justamente la contrapartida del procesamiento secuencial. En suma, esta es una forma altamente eficiente desde el punto de vista del costo, de mejorar la performance de un sistema mediante actividades concurrentes en el computador.

1.2.2.2. - **Utilización conjunta de recursos**

Puede ser que varias actividades concurrentes tengan que compartir determinados recursos o información. Hay cuatro razones para ello :

- el costo; es absurdo disponer de infinitos recursos.
- la posibilidad de trabajar a partir de lo que hicieron otros.
- la posibilidad de compartir datos.
- la eliminación de información redundante.

Asociado a esto tenemos los problemas de ubicar y proteger estos recursos, el acceso simultáneo a la información y la ejecución simultánea de programas.

1.2.2.3. - **Almacenamiento a largo plazo**

Este tipo de almacenamiento permite que el usuario guarde sus datos o programas en el propio computador.

El problema aquí es el de proporcionar un acceso fácil a estos datos, la protección de la información y el resguardo de la misma ante fallas del sistema.

1.2.2.4. - **Indeterminismo**

Un Sistema Operativo debe ser determinista en el sentido de que el mismo programa ejecutado con los mismos datos ayer u hoy debe producir los mismos resultados. Por otro lado es indeterminista en el sentido de que debe responder a circunstancias que pueden ocurrir en un orden impredecible.

Debido a la gran cantidad de eventualidades que pueden darse, es evidentemente poco razonable esperar poder escribir un sistema operativo que las considere todas una a una. En lugar de ello, el sistema debe escribirse de forma que sea capaz de tratar cualquier secuencia de circunstancias de este tipo.

1.2.3. - **Características Deseables**

1.2.3.1. - **Eficiencia**

Si bien la eficiencia de un S.O. es muy importante lamentablemente es difícil establecer un criterio único por el cual pueda juzgarse. Algunos criterios posibles son :

- Tiempo transcurrido entre tareas
- Tiempo ocioso del procesador central
- Tiempo de ejecución de las tareas batch
- Tiempo de respuesta en los sistemas interactivos
- Utilización de los recursos
- Rendimiento (tareas ejecutadas por hora)

No todos estos criterios pueden satisfacerse simultáneamente, siempre que se privilegia uno de ellos es en detrimento de algún otro.

1.2.3.2. - **Fiabilidad**

Teóricamente un Sistema Operativo debe estar completamente libre de todo tipo de errores y ser capaz de resolver satisfactoriamente todas las contingencias que pudiesen presentársele. En la práctica ello nunca ocurre, aunque en este punto se han logrado notables avances.

1.2.3.3. - **Facilidad de corrección**

Debería ser posible corregir o mejorar un S.O. sin tener que hacer uso de todo un ejército de programadores. Esto se puede lograr si el sistema es de construcción modular con interfases claramente definidas entre los diferentes módulos.

La idea aquí es lo que se suele denominar **Sistema Abierto**, al cual, por ejemplo es fácil agregarle drivers (programas especiales para manejar ciertas situaciones o periféricos) que se toman en el momento de la carga o porque sus módulos son sencillos de linkeditar en un único código objeto.

1.2.3.4. - **Tamaño pequeño**

El espacio que consume en la memoria el sistema operativo debería esperarse que fuera pequeño ya que cuanto mayor es el mismo provoca que exista una zona mayor no destinada a tareas de los usuarios. Además un sistema grande está más sujeto a errores

1.2.4. - **PROCESOS CONCURRENTES**

Antes de iniciar un estudio más detallado de los sistemas operativos, introduciremos algunos conceptos fundamentales.

1.2.4.1. - **Programas, Procesos y Procesadores**

Consideremos un sistema operativo como un conjunto de actividades cada una de las cuales lleva a cabo una cierta función, como por ejemplo la administración de las E/S. Cada una de estas actividades consiste en la ejecución de uno o más programas que se ejecutarán toda vez que se requiera tal función.

Utilizaremos la palabra **proceso** para referirnos a una actividad de este tipo.

Consideremos a un programa como una entidad pasiva y a un proceso como una entidad activa, un **proceso** consiste entonces, en una secuencia de acciones llevadas a cabo a través de la ejecución de una serie de instrucciones (un programa), cuyo resultado consiste en proveer alguna función del sistema. Las funciones del usuario también pueden asimilarse a este concepto, es decir que la ejecución de un programa de un usuario también será un proceso.

Un proceso puede involucrar la ejecución de más de un programa. Recíprocamente, un determinado programa o rutina pueden estar involucrados en más de un proceso. De ahí que el conocimiento del programa en particular que está siendo ejecutado no nos diga mucho acerca de la actividad que se está llevando a cabo o de la función que está siendo implementada. Es fundamentalmente por esta razón por lo que es más útil el concepto de proceso que el de programa.

Un proceso es llevado a cabo por acción de un agente (unidad funcional) que ejecuta el programa asociado. Se conoce a esta unidad funcional con el nombre de **procesador**.

Los conceptos de proceso y de procesador pueden emplearse con el fin de interpretar tanto la idea de concurrencia como la de no-determinismo. La concurrencia puede verse como la activación de varios procesos a la vez. Suponiendo que haya tantos procesadores como procesos esto no reviste inconveniente alguno. Pero, si sucede como habitualmente que los procesadores son menos que los procesos se puede lograr una concurrencia aparente conmutando los procesadores de uno a otro proceso. Si esta conmutación se lleva a cabo en intervalos lo suficientemente pequeños, el sistema aparentará un comportamiento concurrente al ser analizado desde la perspectiva de una escala mayor de tiempo.

Resumiendo, un proceso es una secuencia de acciones y es, en consecuencia, dinámico, mientras que un programa es una secuencia de instrucciones y es, así pues, estático. Un procesador es el agente que lleva a cabo un proceso. El no-determinismo y la concurrencia pueden describirse en términos de interrupciones de procesos entre acciones (imposibilidad de prever el momento en el que se produce la interrupción) y de conmutación de procesadores entre procesos (se reparte el tiempo de CPU en rebanadas asignadas a cada proceso). Con el fin de que pueda llevarse a cabo esta conmutación, debe guardarse suficiente información acerca del proceso con el fin de que pueda reemprenderse más tarde.

1.2.4.2. - **Comunicación entre procesos**

Los distintos procesos dentro de un computador no actúan, evidentemente, de forma aislada. Por un lado deben cooperar con el fin de alcanzar el objetivo de poder ejecutar las tareas de los usuarios, y compiten por el uso de los diferentes recursos como ser memoria, procesador y archivos.
Estas dos actividades llevan asociada la necesidad de algún tipo de comunicación. Las áreas en las que esta comunicación es esencial pueden dividirse en :

1.2.4.2.1. - **Exclusión Mutua**

Existen recursos en un sistema de tipo compartibles (es decir pueden ser usados por varios procesos en forma concurrente) o no-compartibles (o sea que su uso se ve restringido a un solo proceso por vez). El hecho de no ser compartible puede deberse a cuestiones de imposibilidad técnica o por la razón de que es necesario su uso individual en virtud de que algún otro proceso pueda interferir con el que lo está usando.

El problema que trata la exclusión mutua es asegurar que los recursos no-compartibles sean accedidos por un solo proceso a la vez.

1.2.4.2.2. - **Sincronización**

En términos generales la velocidad de un proceso respecto a otro es impredecible ya que depende de la frecuencia de la interrupción asociada a cada uno de ellos y de cuán a menudo y por cuánto tiempo tuvo asignado el recurso procesador. Se dice entonces, que un proceso se ejecuta **asincrónicamente** respecto de otro.

Sin embargo existen ciertos instantes en los cuales los procesos deben sincronizar sus actividades. Son estos puntos a partir de los cuales un proceso no puede continuar hasta tanto se haya completado algún tipo de actividad. El Sistema Operativo debe proveer mecanismos que permitan poder llevar a cabo esta sincronización.

1.2.4.2.3. - **Deadlock - Abrazo Mortal**

Cuando varios procesos compiten por los recursos es posible que se de una situación en la cual ninguno de ellos puede proseguir debido a que los recursos que necesita están ocupados por otro proceso. Esta situación se conoce con el nombre de **deadlock**. El evitarlos o al menos limitar sus efectos, es claramente una de las funciones de los sistemas operativos.

Hasta aquí hemos tratado de dar un panorama general sobre lo que son los sistemas operativos. Entrando más de lleno en el temario pasamos ahora a desarrollar los conceptos de arquitecturas.

1.3. - **ARQUITECTURA DE COMPUTADORES**

1.3.1. - **Algunas Definiciones**

Veamos cuales son algunas de las definiciones respecto de sobre qué es la arquitectura de un computador :

- Arte de diseñar una máquina con la cual sea agradable trabajar ". (Caxton Foster - 1970)
- Determinar componentes, funciones de los componentes y reglas de interacción entre ellos " (N. Prassard - 1981)
- La estructura de la Computadora que el programador necesita conocer con el objeto de escribir programas en lenguaje de máquina correctos " (Informe final del proyecto CFA (Arquitectura de Familias de Computadoras).

De acuerdo a esto, cada cual define la Arquitectura de un computador, como ya es obvio, desde su punto de vista e interés de estudio.

La última definición hace hincapié en la interfase que el hardware le presenta al programador.

La segunda hace referencia a la estructura, organización, implementación y comportamiento del computador.

Es evidente que ninguna de las tres definiciones por sí solas dan un entendimiento cabal de lo que es una Arquitectura y aún aceptando que ambas se complementen, faltan aspectos de algoritmos y estructuras lógicas para completar la definición, como el que correspondería a como se ejecuta en realidad una instrucción.

Ténganse en cuenta que :

- Estructura : hace referencia a la interconexión entre los distintos componentes. Por ejemplo que el diseño tenga una CPU, una memoria y un canal de E/S y que la CPU se conecte entre la memoria y el canal de E/S.
- Organización : hace referencia a las interacciones dinámicas y de administración de los componentes. Siguiendo el ejemplo anterior la CPU actuará como comunicación entre las E/S que se realicen cargando los datos transferidos desde el canal a la memoria, asimismo atenderá las señales que se envíen desde el canal y enviará comandos al mismo.
- Implementación : hace referencia al diseño específico de los componentes. Por ejemplo puede ser el tipo de acceso a memoria en cuanto a si es por dirección o por contenido, para lo cual el hardware necesario difiere sensiblemente.

Es válido recordar que la implementación distingue dos computadoras distintas pero de igual arquitectura (Ejemplo : Arquitectura secuencial con o sin Pipeline; en el primer caso la arquitectura requiere de un hardware específico que es el pipeline).

En definitiva una arquitectura está dada por sus :

- Componentes
- Interconexión de sus componentes
- Interacción entre sus componentes
- Implementación de sus componentes
- La forma de usar todos estos aspectos en beneficio de una tarea o sea su **operación** (qué y cómo hace cada instrucción) aliviando o **no** al software en su tarea (microcódigos, subsistema de E/S, etc.).

1.3.2. - **NIVELES DE ARQUITECTURAS**

Teniendo en cuenta los puntos anteriores se puede hablar de niveles de arquitectura :

- **Exoarquitectura** : Es la estructura y capacidad funcional de la arquitectura visible al programador de assembler (qué y cómo hace cada instrucción, por ejemplo una instrucción MVCL -move character long- se utiliza para mover un string de caracteres de longitud variable).

- **Endoarquitectura** : Es la especificación de las capacidades funcionales de los componentes físicos, las estructuras lógicas de sus interconexiones, las interacciones, los flujos y controles de flujos de información (el programador no tiene por que saber que la instrucción MVCL requiere de un número de ciclos de lectura/grabación desde/hacia memoria no determinado a priori y que depende de la longitud del string que se desea mover).

- **Microarquitectura** : Qué componentes se abren o cierran durante la ejecución de una instrucción, en especial si tenemos una unidad de control microprogramada (en el caso MVCL se arranca un microprograma que realiza la transferencia de porciones exactas del string hasta agotar la totalidad del mismo).

Las relaciones entre estos niveles responden a la siguiente figura :

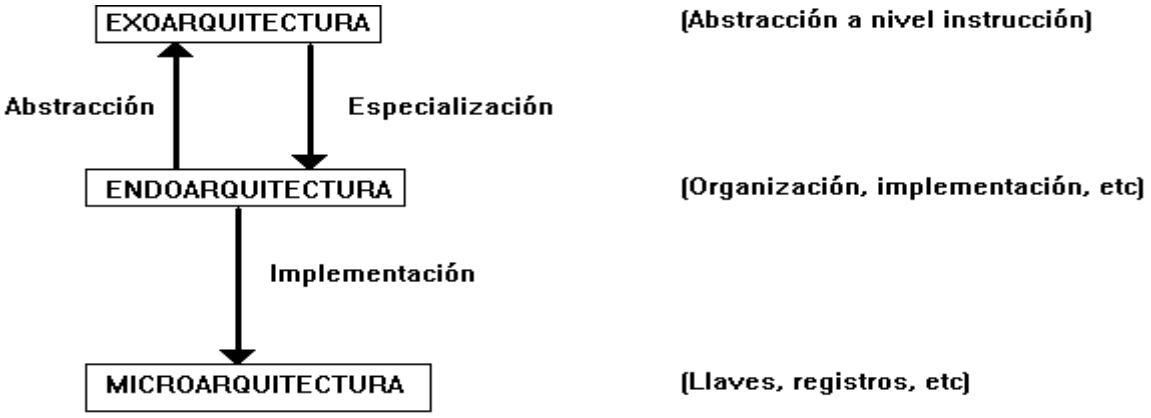

Fig. 1.1. - Niveles de Arquitecturas

1.3.3. - **Breve cuadro evolutivo**

En el siguiente cuadro mostramos muy brevemente la evolución de las diferentes Generaciones de computadoras.

Hay que destacar que el gran salto evolutivo entre la 2da y 3ra Generación está dado además por la presencia de Sistemas Operativos cada vez más potentes y entre la 3ra y la 4ta Generación por mecanismos que permitieron el mejor aprovechamiento de los recursos, como por ejemplo la memoria virtual.

La aparición de la inteligencia distribuida (workstation) y la existencia de redes de computadoras con muchas de sus funciones distribuidas a lo largo de la red están tal vez indicando la aparición de la 5ta Generación, la cual es más claramente entrevista con máquinas implementadas sobre mecanismos de inferencia.

	G E N E R A C I O N E S			
	1era	2da	3ra	4ta
Procesador	Válvulas Toro magnético	Transistor Disco magnético	ICs SSI MSI	LSI VLSI memorias de semiconductores
Estructura	Monoprocesador	Unidades multifunción Proces. E/S	Microprocesadores MIMD (16 proc.) SIMD (64 proc vectoriales)	Workstation LAN SIMD (1024 proc.)
Velocidad	$5 * 10^{-4}$ MFLOP	$5 * 10^{-2}$ MFLOP	1 MFLOP	5 hasta 10000 MFLOP
Unidad de control	Hardwired	Hardwired	Hardware microprogramada	Hardware microprogramada
Caracteris. hardware	Aritmétic. punto fijo	Aritmética de punto flotante	Microprog. Pipeline Mem. Cache	———
Caracteris. software	Lenguaje máquina assembler No hay SO.	Leng. alto nivel. Monitores batch	Multiprogram. Mem. Virtual Sist. Operativ.	Multiprocesamiento

Fig. 1.2. - Evolución de generaciones de computadoras.

1.4. - **ARQUITECTURAS SECUENCIALES**

Casi todas las computadoras están regidas por una misma estructura general, que es la determinada por Von Neumann (1945) y presentada en su informe para su nueva computadora, la EDVAC (Computadora Electrónica de Variable Discreta).

Los conceptos de ese informe de aproximadamente 100 páginas rigen casi todos los diseños de las computadoras actuales, y son :

- Programa almacenado y Datos almacenados (esto significa que tanto programa como datos deben residir en una memoria, común o no, pero que deben estar representados internamente para poder ser ejecutados y tratados).
- Flujo de cómputo secuencial
- Representación binaria de la información (o sea, que datos y programas se representan de igual forma)
- Flujo de información interno en paralelo, en vez de en serie. (esto claramente indica que la cantidad de circuitos debe aumentar hasta tantos bits en paralelo como se pretenda que se muevan juntos en un flujo de datos).
- No paralelismo de las operaciones.

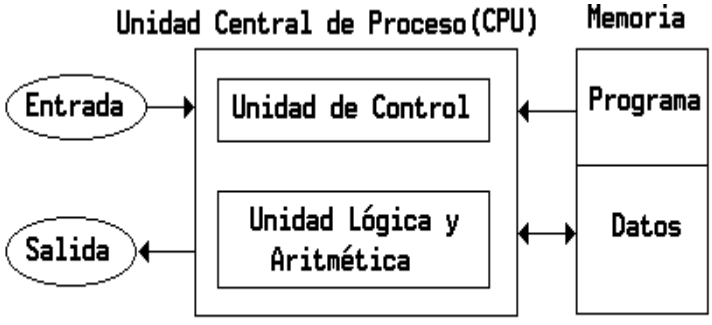

Fig. 1.3. - Arquitectura Von Neumann (1945).

Von Neumann describió cómo una computadora comprende cinco unidades (una de entrada, una de salida, un procesador aritmético, una unidad de control y una memoria) y de qué manera, siguiendo

el criterio de la unidad de control, las instrucciones se pueden almacenar en la misma memoria interna que los datos y pueden ser interpretadas por esa unidad de control de la misma forma que los datos por la unidad aritmética.

1.4.1. - **Computadoras de programa almacenado.**

En la Máquina Analítica (Babbage, siglo 19) y en sus sucesores más modernos como el Mark I de Harvard y en el ENIAC, los programas y los datos se almacenaban en memorias externas separadas.
Ingresar o alterar los programas era una tarea muy tediosa.

La idea de almacenar los programas y los datos en la misma unidad de memoria, el llamado concepto de programa almacenado, se atribuye usualmente a los diseñadores del ENIAC, y más notoriamente al matemático nacido en Hungría John Von Neumann (1903-1957) quien fue un consultor en el proyecto ENIAC.

Además de facilitar la programación, el concepto de programa almacenado hizo posible que un programa modificara sus propias instrucciones.

1.4.2. - **Proceso de Cómputo Secuencial**.

El concepto más difundido cuando se habla de una arquitectura Von Neumann es el del proceso de cómputo.

En estas arquitecturas el programa se almacena en memoria como una secuencia de instrucciones y se ejecuta extrayendo las mismas en secuencia e interpretándolas. Por lo tanto el curso que sigue el cómputo viene dado por la secuencia de las instrucciones del programa (es decir, el flujo de control del programa).

Nótese que uno de los elementos característicos de toda arquitectura Von Neumann es el llamado Registro de Próxima Instrucción (del inglés Program Counter, Contador de Programa) cuya misión consiste en indicar en todo momento cuál es la siguiente instrucción que deberá ejecutarse respetando la secuencia.

1.4.3. - **Arquitectura Básica de un monoprocesador**

Un típico monoprocesador cuenta con tres componentes básicos: la memoria principal, la unidad central de proceso (CPU, que generalmente cuenta con por lo menos una unidad aritmético-lógica -UAL- y una unidad de control), y el subsistema de E/S (con unidades para entrada y para salida o para entrada/salida).

Según John Backus (Agosto 1978), en su forma más simple una computadora Von Neumann consta de tres partes: una unidad central de proceso (CPU), una unidad de almacenamiento y un tubo de conexión que puede transmitir una sola palabra entre la CPU y el almacenamiento (y enviar una dirección desde almacenamiento).

Denominemos a este tubo el *Cuello de botella de Von Neumann.*

El trabajo de un programa consiste en alterar el contenido del almacenamiento de alguna forma; cuando uno piensa que dicha tarea debe realizarla íntegramente bombeando a través del cuello de botella de Von Neumann se da cuenta inmediatamente de la razón de tal nombre.
Irónicamente, una gran parte del tráfico en el cuello de botella no son datos útiles, sino más bien nombres de datos (direcciones), así como operaciones y otros datos que sirven solamente para obtener o calcular tales nombres (instrucciones).

Antes de que un dato sea enviado por el tubo hacia la CPU su dirección debe estar en la CPU; esa dirección llegó allí porque fue enviada por el tubo desde el almacenamiento hacia la CPU o bien, fue generada por la CPU a través de alguna operación.

Si la dirección proviene del almacenamiento entonces su dirección debió provenir del almacenamiento o fue generada por la CPU, y así siguiendo.

Si, por otra parte, la dirección fue generada en la CPU, debió ser generada por alguna regla FIJA (por ejemplo: sumar 1 al contador de programa) o por una instrucción que provino del almacenamiento a través del tubo, en cuyo caso su dirección debió provenir ..... y así siguiendo.

1.4.4. - **Unidades funcionales**

El objetivo de todo esto, o sea el objetivo de una computadora, es que un algoritmo, escrito en una determinada codificación, que constituye el programa y cuyas instrucciones están almacenadas en memoria, instruyan a la Unidad de Control, tomen datos y los transformen en resultados.

De aquí resulta bien claro cuáles son las funciones de las Unidades Funcionales, o sea :

- En la memoria se almacenan datos y programas, sus instrucciones informan a la Unidad de Control qué hacer, por ejemplo, que tome un dato de la Unidad de Entrada lo almacene en memoria, lo lleve al procesador, lo transforme por medio de la Unidad Aritmético/Lógica, lo almacene nuevamente en memoria ya transformado y lo devuelva al medio externo por una Unidad de Salida.
- Las instrucciones son tomadas una a una y en secuencia, obviamente existen instrucciones para alterar esa secuencia, pero es exclusivamente para comenzar una nueva.

En la Fig. 1.4 podemos ver la estructura general de la típica computadora de la serie IBM S/360.

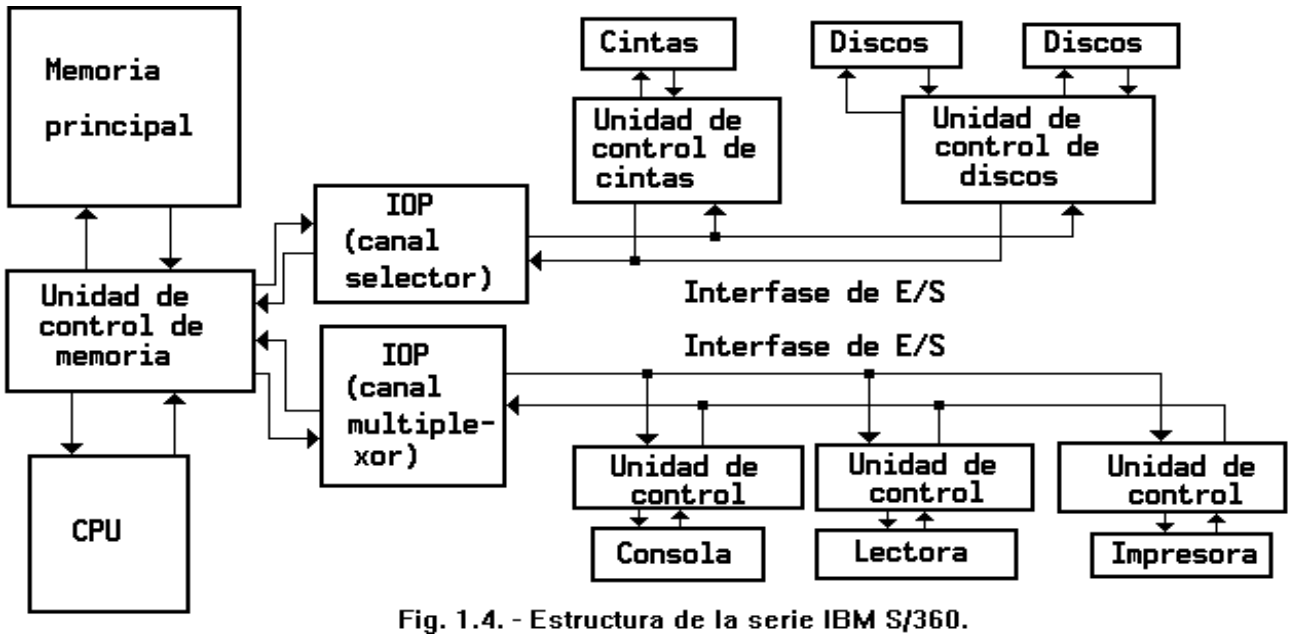

Fig. 1.4. - Estructura de la serie IBM S/360.

La misma utiliza dos tipos de Procesadores de E/S : canales multiplexores y canales selectores.

Los canales multiplexores pueden intercalar (multiplexar) la transmisión de datos entre memoria principal y diferentes dispositivos de E/S, en tanto que sólo un dispositivo de E/S conectado a un canal selector puede transmitir o recibir información en un momento dado (Ver Capítulo 3).

Los canales selectores se utilizaron para dispositivos de muy alta velocidad, por ejemplo : cintas o discos magnéticos, en tanto que los multiplexores sirven para dispositivos de baja velocidad (impresoras, lectoras de tarjetas, etc).

Actualmente se utilizan los canales block multiplexor para dispositivos de muy alta velocidad.

Cada procesador de E/S se encuentra conectado a un bus que se denomina la **interfase de E/S** compuesta por un conjunto de líneas de datos y líneas de control. Este bus de interfase es compartido por todos los periféricos que pertenecen a un determinado procesador de E/S.

Un dispositivo de E/S o un conjunto de dispositivos de E/S del mismo tipo está supervisado localmente por una unidad de control que es particular para el tipo de dispositivo en cuestión.

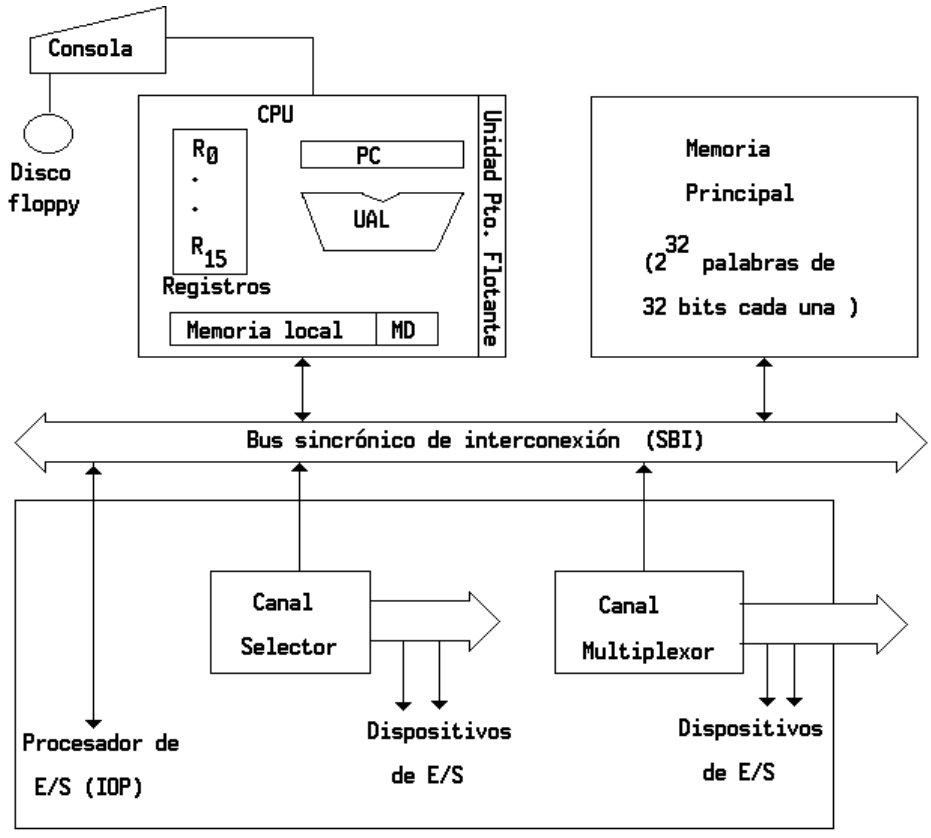

Fig. 1.5. - Arquitectura de sistema de la supermini Vax-11/780 monoprocesador.

En las Fig. 1.5 y 1.6 se pueden observar dos arquitecturas típicas de monoprocesadores.

En la figura del monoprocesador VAX puede verse que:

La CPU contiene :

- 16 registros de uso general, uno de ellos (register status) registra el estado actual del procesador y del programa que está siendo ejecutado; además existe un registro adicional que sirve como la PC (palabra de control) del programa,.
- Una unidad Aritmética y Lógica con punto flotante.
- Una memoria cache local con una memoria de diagnóstico que es optativa.

La CPU, la memoria principal y el subsistema de E/S están todos conectados a un bus sincrónico de interconexión, el synchronous backplane interconnect (SBI).

A través de este bus todos los dispositivos de E/S se pueden comunicar entre sí, con la CPU o con la memoria.

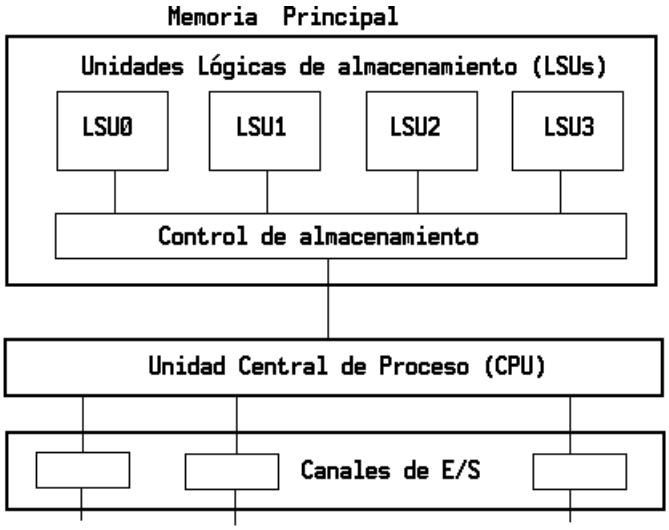

Fig. 1.6. - Arquitectura del mainframe IBM S/370.

Los periféricos pueden estar conectados a este bus por medio de canales selectores o multiplexores o a través de Procesadores específicos de E/S.

En la arquitectura del sistema 370 :

- La CPU contiene el decodificador de instrucciones (normalmente dentro de la Unidad de Control), la unidad de ejecución y también una memoria cache.
- La memoria principal está dividida en cuatro unidades llamadas Unidades Lógicas de Almacenamiento (Logical Storage Unit LSU), que pueden accederse en forma alternada. El controlador de almacenamiento provee una conexión de tipo multiple-port (multipuerta) entre la CPU y las 4 LSU.
- Los periféricos están conectados al sistema vía canales de alta velocidad que operan en forma asincrónica con la CPU.

En la Fig. 1.7 podemos apreciar, por un lado cómo mejoran las velocidades (tomando como parámetro que la velocidad del modelo 30 es 1 el modelo 70 lo

Subsistema	Parámetros	Modelo 30	Modelo 70
Memoria Principal	Ciclo de Memoria	2μs	1μs
	Ancho del bus	8 bits	64 bits
	Velocidad máxima de transferencia	$4 * 10^6$ bits/s	$128 * 10^6$ bits/s
CPU	Ciclo de CPU	30 ns	6 ns
	Tecnología de los registros de trabajo	Toro	Semiconductor
	Ancho del bus de CPU	8 bits	64 bits
	Velocidad de cómputo relativa	1	50

Fig. 1.7. - Comparativa de los parámetros de diseño de dos modelos de la serie IBM S/360.

aventaja 50 veces) respecto de dos modelos de la serie S/360 antes mencionada. Pero, por otra parte, es interesante observar como existe una descompensación entre la velocidad de la memoria y la velocidad de la CPU. Es esta diferencia de velocidades lo que nos permitirá más adelante el analizar los mecanismos mediante los cuales se busca equilibrar esta diferencia de velocidades de los componentes de un computador (Bandwidth).

En todas las arquitecturas se hace necesario lograr un balance entre los distintos subsistemas a efectos de evitar los cuellos de botella y mejorar el THROUGHPUT total del sistema (que es la cantidad de trabajos realizados

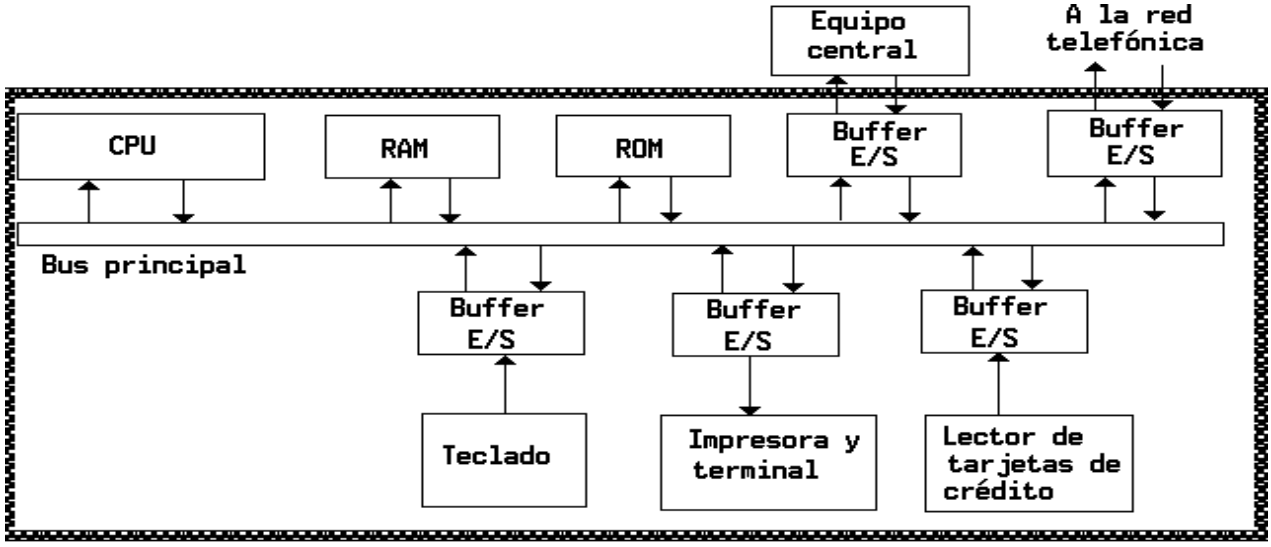

Fig. 1.8. - Computador de propósito específico Punto de Ventas (Point-of-Sales).

por unidad de tiempo).

Existe un número bastante grande de arquitecturas diferentes de acuerdo a la aplicación a la cual se destina el computador en sí. Por ejemplo en la Fig. 1.8 se aprecia la arquitectura de un computador de propósito específico. En este caso en particular se trata un microcomputador de tipo POS (Point-Of-Sales o Punto de ventas) que se han popularizado mucho en los últimos años a raíz de las redes de teleprocesamiento que permiten que un comerciante tenga un instrumento que le permite verificar y actualizar sus movimientos de ventas a clientes.

1.5. - **IMPLEMENTACIÓN REAL DE LAS UNIDADES FUNCIONALES DE UN COMPUTADOR**

1.5.1. - **PROCESADOR O CPU**

Se define la CPU o procesador como la unidad funcional que ejecuta las instrucciones de una determinada arquitectura de propósito general. Muchos sistemas de computación tienen un procesador que se encarga de la totalidad de las funciones de interpretación de instrucciones y su ejecución.

El término de propósito general sirve para diferenciar a la CPU de otros procesadores tales como los procesadores de E/S cuyas funciones son en cierta forma más restringidas o para un propósito específico.

Generalmente existe una sola CPU en la mayoría de las computadoras. Una computadora con una sola CPU se denomina Monoprocesador; una computadora con más CPUs se denomina un Multiprocesador.

Cualquier CPU muestra una bien clara división entre lo que es el procesamiento de los datos y el control del procesamiento en sí. La primer función suele asignársele a una Unidad Aritmético y Lógica (UAL o ALU en inglés), en tanto que la segunda pertenece a la Unidad de Control de Programa o Unidad de Instrucciones.

La Fig. 1.9 muestra un bloque sencillo que se utiliza para representar a la CPU. Existe un bus de datos de una sola palabra que es el camino por el cual fluye la información desde o hacia la CPU.

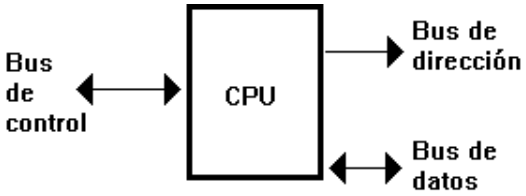

Fig. 1.9. - Una Unidad Central de Proceso.

Generalmente existe además un segundo bus por el cual se transmiten las direcciones desde la CPU a la memoria principal y también a los dispositivos de E/S.

Existen además algunas líneas de Control que se utilizan para controlar los otros componentes del sistema y además para sincronizar sus operaciones con las de la CPU.

La Fig. 1.10 muestra la estructura de la CPU del sistema S/360-370. La unidad aritmético y lógica está dividida en tres subunidades que cumplen las siguientes funciones :

- Operaciones de punto fijo, que incluye aritmética entera y cálculo efectivo de las direcciones.
- Operaciones de punto flotante.
- Operaciones de longitud variable, que incluye aritmética decimal y operaciones sobre strings de caracteres.

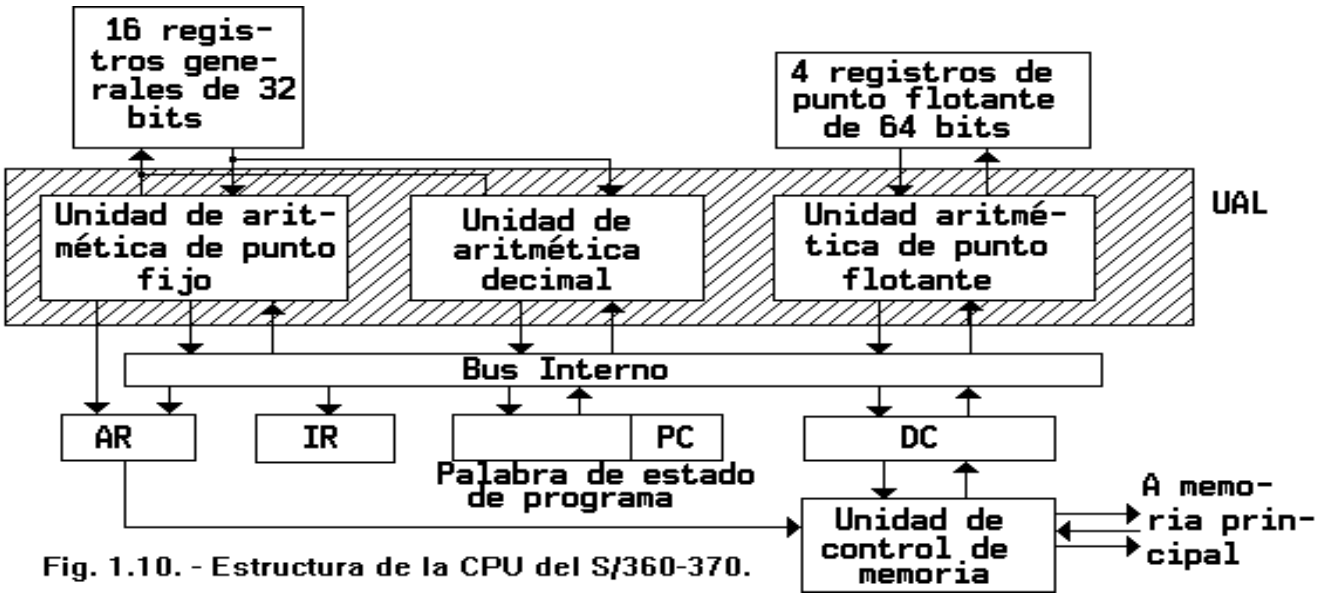

Fig. 1.10. - Estructura de la CPU del S/360-370.

Existen dos conjuntos (cadenas) independientes de registros para almacenar los datos y las direcciones. Los 16 registros generales se usan para almacenar los operandos y los resultados y además pueden usarse de índices. Los 4 registros de punto flotante se usan para la aritmética de punto flotante.

Los registros DR (registro de dato), AR (registro de dirección) e IR (registro de instrucción) son standard.

La Palabra de estado de programa (Program Status Word PSW) almacenada en un registro especial indica el estado del programa, las interrupciones a la cuales la CPU puede responder, y la dirección de la próxima instrucción que se deberá ejecutar (almacenada en el PC -program counter- o registro de próxima instrucción). La razón primordial de la existencia de la PSW es por el manejo de las interrupciones.

La CPU se encuentra en un momento dado en alguno de varios estados. Cuando ejecuta una rutina del sistema operativo, es decir que es el sistema operativo el que tiene el control explícito sobre la CPU, se dice que se encuentra en estado **supervisor** (o modo maestro). Ciertas instrucciones solo
pueden ejecutarse en esta modalidad. Generalmente la CPU se encuentra en estado **problema** (o modo esclavo) mientras ejecuta programas de los usuarios. El estado de la CPU se refleja en la PSW.

La PSW contiene además una clave que se utiliza como **protección de memoria**. La memoria principal está dividida en bloques (de por ejemplo 2K), cada uno de los cuales tiene asignado un número de clave. La clave de almacenamiento especifica el tipo de operación que está permitido realizar sobre ese bloque (lectura, lectura y grabación, ni lectura ni grabación). Una operación que requiera un acceso a un determinado bloque de memoria se ejecuta solamente si la clave de protección del bloque que se desea acceder coincide con la almacenada en la PSW de ese instante.

1.5.2. - **MEMORIA**

La memoria de un sistema de computación se divide en dos mayores subsistemas :

- La Memoria Principal, que consiste en dispositivos relativamente veloces conectados directamente y controlados por la CPU.
- La Memoria Secundaria, con dispositivos de menor velocidad y de menor costo que se comunican indirectamente con la CPU (o no) a través de la memoria principal.

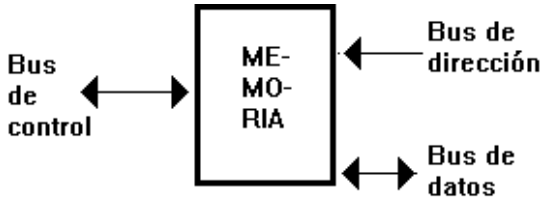

Fig. 1.11. - Una Unidad de Memoria.

Los dispositivos de memoria secundaria tales como discos magnéticos o cintas magnéticas se
utilizan para almacenar grandes cantidades de información. Muy a menudo se encuentran bajo el control directo de procesadores de propósito específico (IOPs input-output processor, procesadores de E/S). Esta memoria secundaria se considera generalmente como parte del subsistema de E/S.

En muchos casos la memoria está organizada en base a palabras direccionables. Esto significa que la información puede ser accedida de a una palabra por vez. La ubicación de cada palabra tiene asociada una dirección que la identifica unívocamente.

Para acceder a una palabra en particular la CPU envía su dirección y una apropiada función de control (Leer o Grabar) a la memoria. Si la función es de grabación además la CPU debe colocar la palabra que desea grabar en el bus de datos de la memoria desde donde será transferida a su correcta ubicación.

1.5.2.1. - **Organización de la Memoria**

Por ahora supondremos que la memoria está compuesta por celdas direccionables y que por cada dirección se acceden n bits (palabras) (8 ó ... 64 ó ...etc) donde no es posible diferenciar instrucciones de datos.

El procesador las diferenciará y las tratará como instrucciones si la dirección proviene de la PC, y como datos si la dirección proviene del operando de la instrucción.

1.5.2.2. - **Formato de Instrucciones**

Los formatos de las instrucciones pueden ser varios, veremos rápidamente algunos para el ejemplo C = A + B.

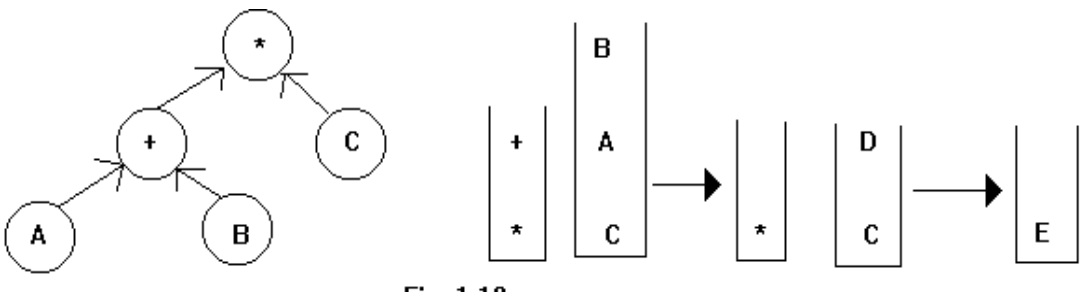

Fig. 1.12.

4 op. ADD A, B, C, D (con D dirección de la próxima instrucción)
3 op. ADD A, B, C
2 op. MOVE C, B Mueve B a C / ADD C, A El resultado queda en C
1 op. LOAD A Carga en acum. o reg. / ADD B Suma Acumulador + B
STORE C Pone el resultado del acumulador en C
0 op. Trabaja con Stacks (Pilas) (sus direcciones están implícitas)
Ejemplo : (A + B) * C

El operador se retira de un stack y los dos operandos del tope del otro stack, y su resultado va al tope del stack de operandos.

1.5.2.3. - **Modos de direccionamiento a Memoria.**

Existen varios tipos de modos de direccionamiento a memoria, aquí veremos rápidamente algunos de ellos.

- Modo Directo : la dirección de los operandos contienen los datos.
- Modo inmediato : el dato está en la instrucción.
- Modo Indirecto : dirección de dirección
- Modo Indexado : dirección + índice, da la dirección

Pueden existir variantes y combinaciones de estos modos de direccionamiento. Además son dependientes de la organización de la memoria, según veremos en el capítulo de Memoria.

1.5.3. - **DISPOSITIVOS DE E/S**

Los dispositivos de E/S son el medio por el cual un computador se comunica con el mundo exterior. La función primaria de la mayoría de los dispositivos de E/S es actuar como traductores, por ejemplo, convirtiendo información de una forma de representación física en otra.

1.5.3.1. - **Interconexión de Unidades Funcionales. Estructura de Bus**

Las unidades funcionales deben estar interconectadas en forma organizada, por medio de cables (vías) que transportan señales, (bits de datos, direcciones, etc) a estos conjuntos de cables se los denomina **buses** (se pueden encontrar en la bibliografía también con los nombres : red de conmutación (switching network), controlador de comunicaciones y controlador de buses).

La función primaria de los buses es establecer una comunicación dinámica entre los dispositivos que se hallan bajo su control.

Por razones de costo los buses se comparten. Esto lleva a producir lo que se denomina **contención** en el bus, cuando más de un componente desea hacer uso del él al mismo tiempo. El bus debe resolver tal contención seleccionando uno de los pedidos en base a algún tipo de algoritmo (por ejemplo, prioridad) y encolando el resto a la espera de su liberación.

Los requerimientos simultáneos a un mismo dispositivo surgen del hecho de que la comunicación entre los diferentes componentes del sistema es generalmente de tipo asincrónica.

Esto se puede atribuir a diversas causas :

- Existe un alto grado de independencia entre los componentes. Por ejemplo, la CPU y los IOPs ejecutan diferentes programas e interactúan frecuentemente en momentos impredecibles.
- Las velocidades de los diferentes componentes varían dentro de un amplio rango. Por ejemplo la CPU opera de a 1 a 10 veces más velozmente que los dispositivos de memoria, y esta a su vez es varias veces más veloz que los dispositivos de E/S.
- La distancia física que separa los componentes puede ser grande de manera tal que sea imposible lograr una transmisión sincrónica de la información entre ambos.

La transmisión asincrónica se implementa muy frecuentemente con un mecanismo que se denomina **"handshaking"** (acuerdo). Supongamos que tenemos que trasmitir un dato desde el dispositivo A hacia el B. A coloca el dato en cuestión en el bus de datos de A a B y envía luego una señal de control hacia B, que se denomina generalmente **"ready"**, para indicarle la presencia del dato en el bus. Cuando B reconoce la señal de ready, transfiere el dato desde el bus hacia un área propia y luego activa una línea de control de **"acknowledge"** (toma conoci
miento) hacia A. Cuando A recibe la señal de acknowledge comienza la transmisión del siguiente dato. Luego una secuencia de ready/acknowledge acompaña la transferencia de los datos haciéndola independiente de las velocidades de los dos componentes.

Las líneas de datos y de control conectadas a un dispositivo así como la secuencia de señales requeridas para comunicarse constituyen la **"interfase"** del dispositivo. Cuanto más similares sean las interfases de los dispositivos mucho más sencillo será que se puedan comunicar. Si se utilizan interfases standard en todo el sistema, se incrementa muchísimo la facilidad con la cual el mismo puede expandirse o modificarse.

Veremos a continuación las tres estructuras más usadas de buses.

1.5.3.2. - **Dos Buses**

Todo dato de E/S pasará por el procesador, luego las operaciones y transferencias de E/S se realizan bajo control directo del procesador, o sea la E/S es controlada por programa.

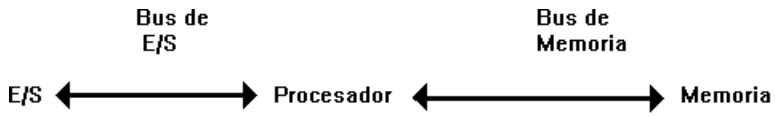

Fig. 1.13.

1.5.3.3. - **Dos Buses (Alternativa) (Canales)**

Aquí las operaciones de E/S y transferencia se realizan bajo control de procesadores de E/S (canales de E/S).

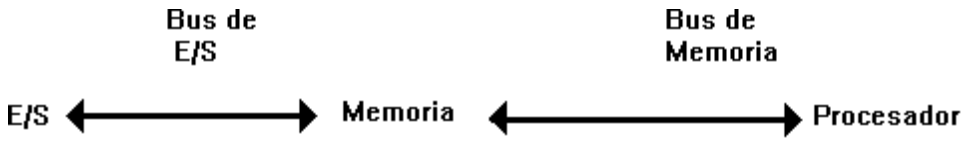

Fig. 1.14.

Generalmente el procesador pasa a los canales los parámetros y el control de las operaciones de E/S.

Aquí surgen los problemas de acceso a memoria simultáneos, los que deben ser resueltos, una forma es por medio de una sola entrada (Fig. 1.15).

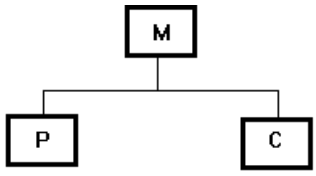

Fig. 1.15.

En este caso se produce el llamado robo de ciclos, o sea el canal roba ciclos de acceso a memoria al procesador.

Puede existir algún control de acceso a memoria tipo MMU (Memory Management Unit) que evite la posibilidad de accesos simultáneos a una misma posición de memoria (Fig. 1.16).

Otra forma de controlar este problema es por medio de la técnica de Interleaving (ver Capítulo 2), también con un controlador de memoria, pero que permite la entrada simultánea a memoria sobre distintas porciones (franjas o bancos).

Aumentando la cantidad de buses obviamente se aumenta la capacidad de transferencia simultánea de información, este caso como los dos anteriores es lo que generalmente se conoce como estructura **Multibus**.

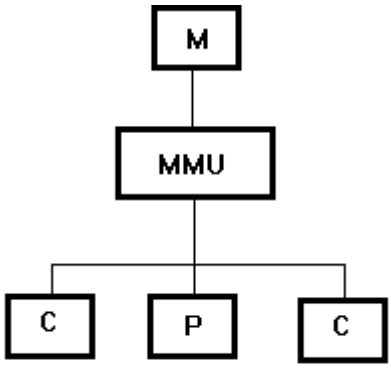

Fig. 1.16.

1.5.3.4. **Un bus**

Aquí el bus se puede utilizar para una sola transferencia por vez (obvio, hay un solo bus) o sea que solo dos unidades funcionales pueden estar haciendo uso de él.

Es fácil incorporar nuevos periféricos.

Es una estructura de bajo costo y es muy utilizada en microcomputadoras.

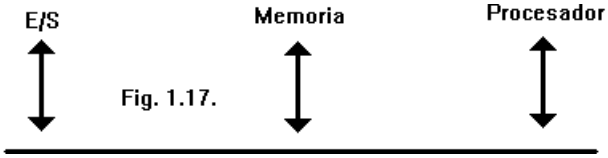

Fig. 1.17.

A pesar de estas diferencias de estructura que hacen al rendimiento, los principios fundamentales de funcionamiento son independientes de ellas y siguen siendo máquinas secuenciales.

1.5.4. - **ORGANIZACION DE PROCESADORES**

1.5.4.1. - **Procesador de un Bus**

En el caso de un procesador de un solo bus la ALU y todos los registros del procesador están conectados a través de un único bus.

En la Fig. 1.18 podemos ver este esquema donde :

RDM : Registro de dirección de Memoria

RBM : Registro buffer de memoria

R_0 a R_n = son registros de uso general

PC = Program Counter

J = es un registro

Z = es un registro acumulador

RI = Registro de instrucción

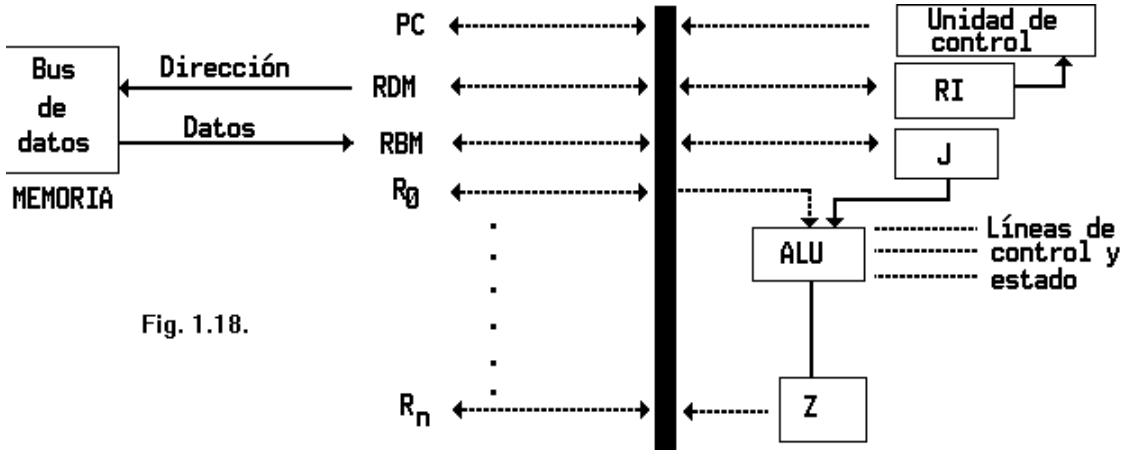

Fig. 1.18.

Para ejecutar la instrucción ADD R1,Op (sumar Op + R1 dejando el resultado en R1) (suponemos instrucciones de igual longitud, y mientras se pone la dirección de lectura, ya se incrementa el RPI) primero se levanta la instrucción propiamente dicha haciendo:

- PC out, RDM in, Read (memoria), Clear J, Carry = 1, ADD, Z in .

Luego la ejecución propiamente dicha será :

- Z out, PC in, Espera (Función de Memoria)
- RBM out, RI in
- Campo Dir RI out, RDM in, Read (memoria)
- R1 out, J in, Espera (función de memoria)
- RBM out, ADD, Z in
- Z out, R1 in

Si lo hacemos para la instrucción ADD R2, R1 (sumar R1 + R2 dejando el resultado en R2) la carga de la instrucción será :

- PC out, RDM in, Read (Memoria), Clear J, Carry = 1, ADD, Z in

y luego, la ejecución :

- Z out, PC in, Espera (Función de Memoria)
- RBM out, RI in
- R2 out, J in
- R1 out, ADD, Z in
- Z out, R2 in

1.5.4.2. - **Procesadores Multibus**

Veamos sobre el ejemplo de la Fig. 1.19 la operación ADD R2, R1.

La carga de instrucción será :

- PC out, G (hab), RDM in, Read (Memoria), J in

y la ejecución entonces :

- Carry = 1, ADD, PC in, Espera (Función de Memoria)
- RBM out, G (hab), RI in
- R1 out, G (hab), J in
- R2 out, ADD, ALU out, R2 in

Con lo cual vemos que esta suma se redujo a un "tiempo" con respecto al anterior, o sea, si tenemos más caminos (buses) más operaciones en paralelo se pueden realizar.

Si por ejemplo se tiene una estructura como la de la PDP-11 (Fig. 1.20).

La ejecución de la operación ADD R3, R2, R1 resultaría en la ejecución de sólo:

- R1 out_A,

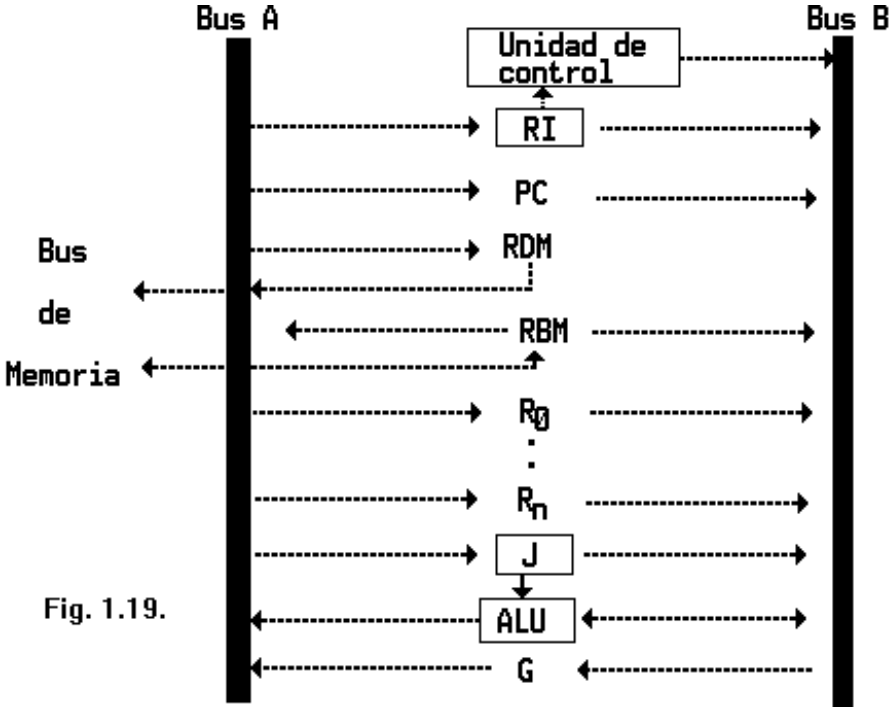

Fig. 1.19.

- A in_A,
- R2 out_B,
- B in_B,
- A out,
- B out,
- ADD,
- ALU out_C,
- R3 in_C

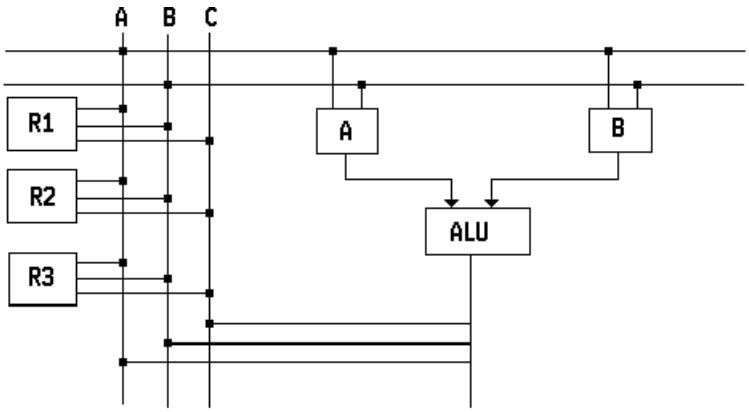

Fig. 1.20. - Estructura de la PDP-11.

1.5.5. - **UNIDAD DE CONTROL (Procesador)**

Si observamos la ejecución de las instrucciones anteriores, veremos que ésta consta de una serie de pasos, en los cuales es necesario dar tiempo para que se completen las operaciones de cada una de ellas.

Estos tiempos (que por razones de simplicidad los tomamos como la unidad y cada paso tarda una unidad) deben ser generados por un contador activado por un reloj.

La unidad de control deberá entonces **no** sólo decodificar un código de operación, sino también emitir señales (códigos de condición) según se vayan ejecutando las instrucciones (overflow, carry -acarreo-, cero, negativo, etc). Es decir la Unidad de Control recibe de la Unidad Aritmético-Lógica señales que indican el estado que ha resultado de la ejecución de la operación indicada.

Nótese por ejemplo que existen ciertas instrucciones que permiten realizar una cierta operación independientemente de si se produce o no overflow; en tales casos al recibir la señal de overflow de la ALU la Unidad de Control no toma ninguna acción, ni de recupero del error (si fuera esto posible) ni indicando que se ha producido una falla debido a que el código de operación de la acción solicitada específicamente indica que tal condición no debe atenderse.

1.5.5.1. - **Microprogramación**

La microprogramación es una técnica para implementar la función de control de un procesador de una manera flexible y sistemática.

El concepto fue enunciado por primera vez por Maurice V. Wilkes en 1951, y fue implementado por primera vez en algunos de los computadores de primera y segunda generación.

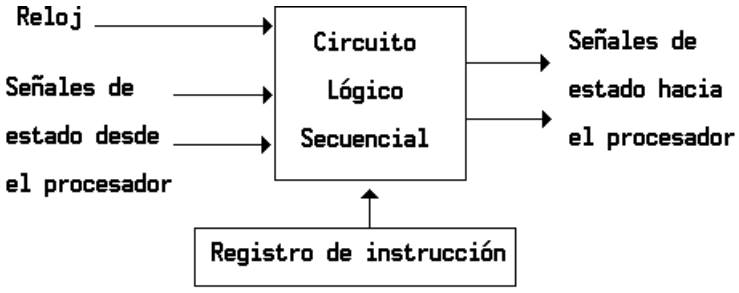

Fig. 1.21. - Circuito de control hardwired.

Sin embargo no fue hasta mediados de los años 60 en los que su aparición en algunos modelos de las series IBM S/360 determinó su mayor difusión.

La microprogramación debe ser considerada como una alternativa del control de circuitos hardwired (o cableado).

Un circuito de control hardwired es típicamente un circuito secuencial como puede verse en la Fig. 1.21. Un circuito de control microprogramado tiene la estructura que puede apreciarse en la Fig. 1.22.

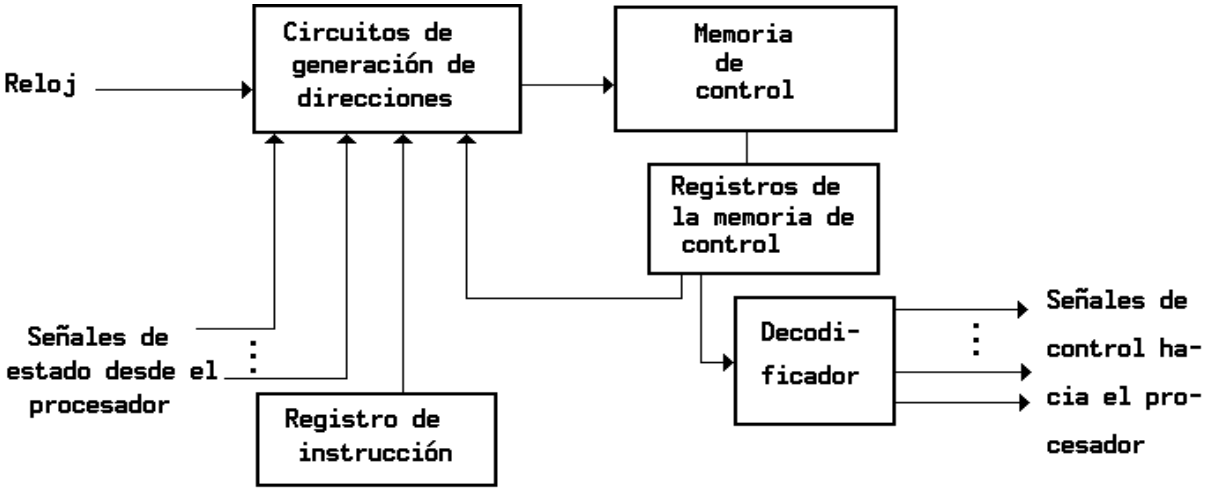

Fig. 1.22. - Circuito de control microprogramado.

En el caso de una unidad de control implementada en hardware será un conjunto de circuitos que interpretan códigos de operación y generan señales dependiendo de los estados de cada instrucción.

En la microprogramación cada instrucción que se encuentra bajo el control del procesador desencadena una secuencia de microinstrucciones llamada microprograma, que se obtienen de una memoria especial de acceso random (al azar) denominada memoria de microprogramas.

Las microinstrucciones indican la secuencia de microoperaciones o las transferencias entre registros necesarias para interpretar y ejecutar la instrucción madre.

Cada instrucción obtenida de la memoria principal del sistema inicia una secuencia de microinstrucciones obtenidas de la memoria de microprogramas.

Para ilustrar su funcionamiento tomemos como ejemplo los 3 últimos pasos de la instrucción ADD R2, R1 (última suma del punto 1.5.4.1))

R1 in	R1 out	R2 in	R2 out	J in	J out	Z in	Z out	ADD...
0	1	0	0	1	0	0	0	0
0	0	0	1	0	0	1	0	1
0	0	1	0	0	0	0	1	0

Donde lo anterior representa los pasos de control para poder ejecutar esa porción de instrucción.

Cada una de las líneas recibe el nombre de microinstrucción y un conjunto de microinstrucciones que corresponden a la resolución de una instrucción, recibe el nombre de microprograma de la instrucción.

La microprogramación provee una forma sistemática para el diseño de circuitos de control, ya que permite estandarizar el control incluyéndolo directamente en el hardware de la máquina deseada.

Potencia grandemente la flexibilidad de una computadora. El conjunto de instrucciones de una máquina microprogramada puede cambiarse simplemente cambiando la memoria de control.

Esto hace posible que una computadora microprogramada ejecute directamente programas escritos en otros lenguajes de máquina o en una computadora diferente, proceso denominado emulación.

Las unidades de control microprogramado tienden a ser más costosas y más lentas que las de tipo hardwired porque la ejecución de una instrucción implica varios accesos a la memoria de microprogramas, o sea este tiempo de acceso es importante en el tiempo de ejecución de una instrucción.

Debido a la estrecha interacción entre el software y el hardware en las unidades microprogramadas se suele referir a los microprogramas como firmware.

EJERCICIOS

1) En base a qué se define o describe una arquitectura ?
2) Cuáles son las unidades funcionales de un computador ?
3) Cuáles son las características distintivas de las 4 grandes generaciones de computadores ?
4) Puede existir concurrencia en un sistema con una única CPU ?
5) Cuál es el nivel de arquitectura que especifica las capacidades funcionales de los componentes físicos de un computador ?
6) Cuál es la función de la UC ?
7) La palabra de estado de programa (PSW) según sus funciones pertenece al componente hardware UC. Justifique el porqué.
8) Cómo funciona un mecanismo de comunicación handshaking ? Es sincrónico o asincrónico ?
9) Cuál es la diferencia entre una UC microprogramada y otra hardwired ?
10) Cómo es una estructura de interconexión de dos buses ?
11) Cuál de las siguientes implementaciones es correcta y porqué :
- Un único bus conecta la memoria, la CPU y los canales
- El acceso a memoria desde los periféricos es siempre a través de la CPU.

12) Cuál es el inconveniente típico de un procesador de un bus común ?
13) Cuáles son los dos conceptos que se manejan cuando uno se refiere a una máquina de arquitectura Von Neumann ?
14) Qué es un microprograma y para qué se utiliza ?
15) Qué es Throughput ?