Tema 1

1.- Soporte hardware (HW) al sistema operativo.

1.1 Elementos básicos

En un nivel alto, un computador consiste de una serie de componentes interconectados de manera que se pueda lograr la función principal del computador, que es ejecutar programas. Por tanto, hay cuatro elementos estructurales principales:

- **Procesador**. Controla el funcionamiento del computador y realiza sus funciones de procesa miento de datos. Cuando sólo hay un procesador, se denomina usualmente unidad central de proceso (Central Processing Unit, CPU).
- **Memoria principal**. Almacena datos y programas. Esta memoria es habitualmente volátil; es decir, cuando se apaga el computador, se pierde su contenido. En contraste, el contenido de la memoria del disco se mantiene incluso cuando se apaga el computador. A la memoria principal se le denomina también memoria real o memoria primaria.
- **Módulos de E/S**. Transfieren los datos entre el computador y su entorno externo. El entorno externo está formado por diversos dispositivos, incluyendo dispositivos de memoria secundaria (por ejemplo, discos), equipos de comunicaciones y terminales.
- **Bus del sistema**. Proporciona comunicación entre los procesadores, la memoria principal y los
  módulos de E/S.

Una de las funciones del procesador es el intercambio de datos con la memoria. Para este fin, se utilizan normalmente dos **registros internos** (al procesador): un registro de dirección de memoria (RDIM), que especifica la dirección de memoria de la siguiente lectura o escritura; y un registro de datos de memoria (RDAM), que contiene los datos que se van a escribir en la memoria o que recibe los datos leídos de la memoria. De manera similar, un registro de dirección de E/S (RDIE/S) especifica un determinado dispositivo de E/S, y un registro de datos de E/S (RDAE/S) permite el intercambio de datos entre un módulo de E/S y el procesador.

Un **módulo de memoria** consta de un conjunto de posiciones definidas mediante direcciones numeradas secuencialmente. Cada posición contiene un patrón de bits que se puede interpretar como una instrucción o como datos. Un módulo de E/S transfiere datos desde los dispositivos externos hacia el procesador y la memoria, y viceversa. Contiene buffers (es decir, zonas de almacenamiento internas) que mantienen temporalmente los datos hasta que se puedan enviar.

1.2 Resgistros del procesador

Un procesador incluye un conjunto de registros que proporcionan un tipo de memoria que es más rápida y de menor capacidad que la memoria principal. Los registros del procesador sirven para dos funciones:

Registros visibles para el usuario.

Permiten al programador en lenguaje máquina o en ensamblador minimizar las referencias a memoria principal optimizando el uso de registros. A un registro visible para el usuario se puede acceder por medio del lenguaje de máquina ejecutado por el procesador que está generalmente disponible para todos los programas, incluyendo tanto programas de aplicación como programas de sistema. Los tipos de registros que están normalmente disponibles son registros de datos, de dirección y de códigos de condición.

- El programador puede utilizar los **registros de datos** para diversas funciones. En algunos casos, son, en esencia, de propósito general y pueden usarse con cualquier instrucción de máquina que realice operaciones sobre datos. Sin embargo, frecuentemente, hay restricciones. Por ejemplo, puede haber registros dedicados a operaciones de coma flotante y otros a operaciones con enteros.
- Los **registros de dirección** contienen direcciones de memoria principal de datos e instrucciones, o una parte de la dirección que se utiliza en el cálculo de la dirección efectiva o completa. Estos registros pueden ser en sí mismos de propósito general, o pueden estar

dedicados a una forma, o modo, particular de direccionamiento de memoria (*ej.*- registro índice, puntero de segmento, puntero de pila).

En algunas máquinas, una llamada a una subrutina o a un procedimiento implica salvar automáticamente todos los registros visibles para el usuario, que se restaurarán al retornar. En otras máquinas, el programador es el responsable de guardar el contenido de los registros visibles para el usuario antes de una llamada a un procedimiento, incluyendo instrucciones para ello en el programa. Por tanto, las funciones de salvar y restaurar se pueden realizar en hardware o en software, dependiendo del procesador.

Registros de control y estado.

Usados por el procesador para controlar su operación y por rutinas privilegiadas del sistema operativo para controlar la ejecución de programas. Se emplean varios registros del procesador para controlar el funcionamiento del mismo. En la mayoría de las máquinas, muchos de ellos no son visibles para el usuario. A algunos de ellos se puede acceder mediante instrucciones de máquina ejecutadas en lo que se denomina modo de control o de sistema operativo.

Por supuesto, diferentes máquinas tendrán distintas organizaciones de registros y utilizarán diferente terminología. Junto los registros *RDIRM, RDAM, RDIE/S* y *RDAE/S*, los siguientes son esenciales para la ejecución de instrucciones:

- **Contador de programa** (Program Counter, PC). Contiene la dirección de la próxima instrucción que se leerá de la memoria.
- **Registro de instrucción** (Instruction Register, IR). Contiene la última instrucción leída.

Otro importante es:

- Los **códigos de condición** (también llamados indicadores) son bits cuyo valor lo asigna normalmente el hardware de procesador teniendo en cuenta el resultado de las operaciones.

No hay una clasificación nítida de los registros entre estas dos categorías. Por ejemplo, en algunas máquinas el contador de programa es visible para el usuario, pero en muchas otras no lo es. Sin embargo, para el estudio que se presenta a continuación, es conveniente utilizar estas categorías.

1.3 Ejecución de instrucciones

Un programa que va a ejecutarse en un procesador consta de un conjunto de instrucciones almacenado en memoria. En su forma más simple, el procesamiento de una instrucción consta de dos pasos: el procesador lee (busca) instrucciones de la memoria, una cada vez, y ejecuta cada una de ellas. La ejecución del programa consiste en repetir el proceso de búsqueda y ejecución de instrucciones. La ejecución de la instrucción puede involucrar varias operaciones dependiendo de la naturaleza de la misma. A estos dos pasos se les denomina fase de búsqueda y de ejecución. La ejecución del programa se detiene sólo si se apaga la máquina, se produce algún tipo de error irrecuperable o se ejecuta una instrucción del programa que para el procesador.

Se denomina **ciclo de instrucción** al procesamiento requerido por una única instrucción.

Búsqueda y ejecución de una instrucción

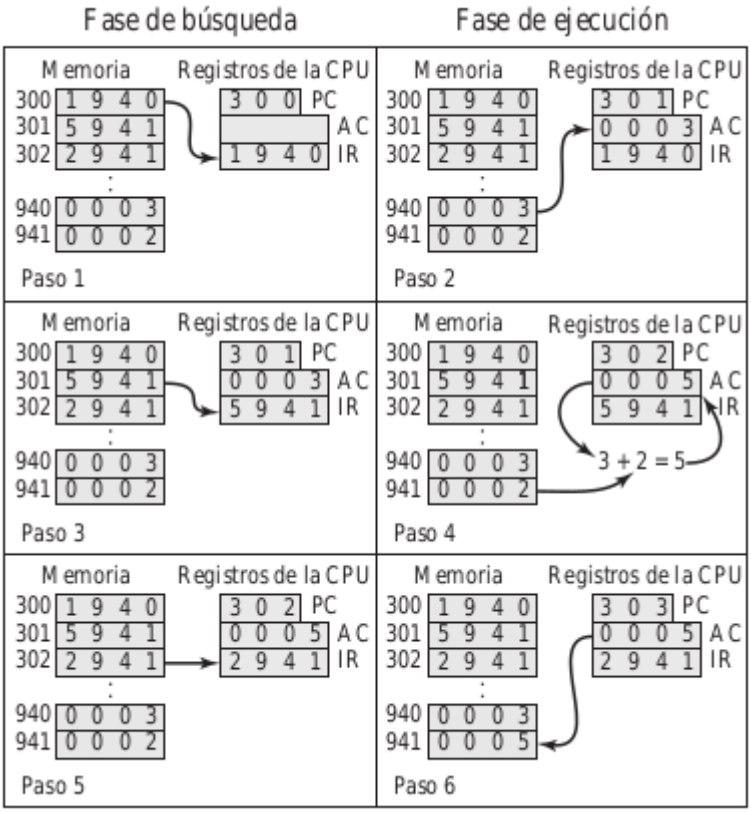
Al principio de cada ciclo de instrucción, el procesador lee una instrucción de la memoria. En un procesador típico, el contador del programa (PC) almacena la dirección de la siguiente instrucción que se va a leer. A menos que se le indique otra cosa, el procesador siempre incrementa el PC después de cada instrucción ejecutada, de manera que se leerá la siguiente instrucción en orden secuencial (es decir, la instrucción situada en la siguiente dirección de memoria más alta).

La instrucción leída se carga dentro de un registro del procesador conocido como registro de instrucción (IR). La instrucción contiene bits que especifican la acción que debe realizar el procesador. El procesador interpreta la instrucción y lleva a cabo la acción requerida. En general, estas acciones se dividen en cuatro categorías:

- **Procesador-memoria**. Se pueden transferir datos desde el procesador a la memoria o viceversa.
- **Procesador-E/S**. Se pueden enviar datos a un dispositivo periférico o recibirlos desde el mismo, transfiriéndolos entre el procesador y un módulo de E/S.
- **Procesamiento de datos**. El procesador puede realizar algunas operaciones aritméticas o lógicas sobre los datos.

- **Control**. Una instrucción puede especificar que se va a alterar la secuencia de ejecución. Por ejemplo, el procesador puede leer una instrucción de la posición 149, que especifica que la siguiente instrucción estará en la posición 182. El procesador almacenará en el contador del programa un valor de 182. Como consecuencia, en la siguiente fase de búsqueda, se leerá la instrucción de la posición 182 en vez de la 150.

Una ejecución de una instrucción puede involucrar una combinación de estas acciones.



1.4 Interrupciones

Prácticamente todos los computadores proporcionan un mecanismo por el cual otros módulos (memoria y E/S) pueden interrumpir el secuenciamiento normal del procesador. Los tipos más comunes de interrupciones son:

- **De programa:** Generada por alguna condición que se produce como resultado de la ejecución de una instrucción, tales como un desbordamiento aritmético o referencias fuera del espacio de la memoria permitido para un usuario.
- **Por temporizador:** Generada por un temporizador del procesador. Permite al sistema operativo realizar ciertas funciones de forma regular.
- **De E/S**: Generada por un controlador de E/S para señalar la conclusión normal de una operación o para indicar diversas condiciones de error.
- **Por fallo del hardware:**Generada por un fallo, como un fallo en el suministro de energía o un error de paridad en la memoria.

Básicamente, las interrupciones constituyen una manera de mejorar la utilización del procesador, ya que, por ejemplo, la mayoría de los dispositivos de E/S son mucho más lentos que el procesador, o el procesador puede dedicarse a ejecutar otras instrucciones mientras que una operación de E/S se está llevando a cabo.

De cara al programa de usuario, una interrupción suspende la secuencia normal de ejecución. Cuando se completa el procesamiento de la interrupción, se reanuda la ejecución. Por tanto, el programa de usuario no tiene que contener ningún código especial para tratar las interrupciones; el procesador y el sistema operativo son responsables de suspender el programa de usuario y, posteriormente, reanudarlo en el mismo punto.

Para tratar las interrupciones, se añade una fase de interrupción al ciclo de instrucción. En la fase de interrupción, el procesador comprueba si se ha producido cualquier interrupción, hecho indicado por la presencia de una señal de interrupción. Si no hay interrupciones pendientes, el procesador continúa con la fase de búsqueda y lee la siguiente instrucción del programa actual. Si está pendiente una interrupción, el procesador suspende la ejecución del programa actual y ejecuta la

rutina del manejador de interrupción. La rutina del manejador de interrupción es generalmente parte del sistema operativo. Normalmente, esta rutina determina la naturaleza de la interrupción y realiza las acciones que se requieran.

Procesamiento de interrupciones

La aparición de una interrupción dispara varios eventos, tanto en el hardware del procesador como en
el software. Cuando un dispositivo de E/S completa una operación de E/S, se produce la siguiente secuencia de eventos en el hardware:

1. El dispositivo genera una señal de **interrupción** hacia el procesador.
2. El procesador **termina la ejecución** de la instrucción actual antes de responder a la interrupción.
3. El procesador comprueba si hay una **petición de interrupción** pendiente, determina que hay una y manda una señal de reconocimiento al dispositivo que produjo la interrupción. Este reconocimiento permite que el dispositivo elimine su señal de interrupción.
4. En ese momento, el procesador necesita prepararse para transferir el control a la rutina de interrupción. Para comenzar, necesita **salvar la información** requerida para reanudar el programa actual en el momento de la interrupción. La información mínima requerida es la palabra de es-
   tado del programa (PSW) y la posición de la siguiente instrucción que se va a ejecutar, que está contenida en el contador de programa. Esta información se puede apilar en la pila de control de sistema.
5. A continuación, el procesador **carga** el contador del programa con la posición del punto de entrada de la rutina de manejo de interrupción que responderá a esta interrupción. Dependiendo de la arquitectura de computador y del diseño del sistema operativo, puede haber un único
   programa, uno por cada tipo de interrupción o uno por cada dispositivo y tipo de interrupción. Si hay más de una rutina de manejo de interrupción, el procesador debe determinar cuál invocar. Esta información puede estar incluida en la señal de interrupción original o el procesador puede tener que realizar una petición al dispositivo que generó la interrupción para obtener una respuesta que contiene la información requerida.

   Una vez que se ha cargado el contador del programa, el procesador continúa con el siguiente ciclo de instrucción, que comienza con una lectura de instrucción. Dado que la lectura de la instrucción está determinada por el contenido del contador del programa, el resultado es que se transfiere el control al programa manejador de interrupción. La ejecución de este programa conlleva las siguientes operaciones:
6. En este momento, el contador del programa y la PSW vinculados con el programa interrumpido se han **almacenado** en la pila del sistema. Sin embargo, hay otra información que se considera parte del estado del programa en ejecución. En concreto, se necesita salvar el contenido de los
   registros del procesador, puesto que estos registros los podría utilizar el manejador de interrupciones, y cualquier otra información de estado. Generalmente, el manejador de interrupción comenzará salvando el contenido de todos los registros en la pila.
7. El manejador de interrupción puede en este momento comenzar a **procesar** la interrupción. Esto incluirá un examen de la información de estado relacionada con la operación de E/S o con otro evento distinto que haya causado la interrupción. Asimismo, puede implicar el envío de mandatos adicionales o reconocimientos al dispositivo de E/S.
8. Cuando se completa el procesamiento de la interrupción, se **recuperan** los valores de los registros salvados en la pila y se restituyen en los registros.
9. La última acción consiste en **restituir** de la pila los valores de la PSW y del contador del programa. Como resultado, la siguiente instrucción que se va ejecutar corresponderá al programa previamente interrumpido.

Es importante salvar toda la información de estado del programa interrumpido para su posterior reanudación. Esto se debe a que la interrupción no es una rutina llamada desde el programa. En su lugar, la interrupción puede suceder en cualquier momento y, por tanto, en cualquier punto de la ejecución de un programa de usuario. Su aparición es imprevisible.

Múltiples interrupciones

Se pueden considerar dos alternativas a la hora de tratar con múltiples interrupciones.

- La primera es inhabilitar las interrupciones mientras que se está procesando una interrupción. Una interrupción inhabilitada significa simplemente que el procesador ignorará cualquier nueva señal de petición de interrupción. Si se produce una interrupción durante este tiempo, generalmente permanecerá pendiente de ser procesada, de manera que el procesador sólo la comprobará después de que se rehabiliten las interrupciones. Esta estrategia es válida y sencilla, puesto que las interrupciones se manejan en estricto orden secuencial, pero la desventaja es que no tiene en cuenta la prioridad relativa o el grado de urgencia de las interrupciones.
- Una segunda estrategia es definir prioridades para las interrupciones y permitir que una interrupción de más prioridad cause que se interrumpa la ejecución de un manejador de una interrupción de menor prioridad.

1.5 Técnicas de comunicación E/S

Hay tres técnicas para llevar a cabo las operaciones de E/S:

E/S programada

Cuando el procesador ejecuta un programa y encuentra una instrucción relacionada con la E/S, ejecuta esa instrucción generando un mandato al módulo de E/S apropiado. En el caso de la E/S programada, el módulo de E/S realiza la acción solicitada y fija los bits correspondientes en el registro de estado de E/S, pero no realiza ninguna acción para avisar al procesador. En concreto, no interrumpe al procesador. Por tanto, después de que se invoca la instrucción de E/S, el procesador debe tomar un papel activo para determinar cuándo se completa la instrucción de E/S. Por este motivo, el procesador comprueba periódicamente el estado del módulo de E/S hasta que encuentra que se ha completado la operación.

Con esta técnica, el procesador es responsable de extraer los datos de la memoria principal en una operación de salida y de almacenarlos en ella en una operación de entrada. El sotfware de E/S se escribe de manera que el procesador ejecuta instrucciones que le dan control directo de la operación de
E/S, incluyendo comprobar el estado del dispositivo, enviar un mandato de lectura o de escritura, y transferir los datos. Por tanto, el juego de instrucciones incluye instrucciones de E/S de las siguientes categorías:

- **Control**. Utilizadas para activar un dispositivo externo y especificarle qué debe hacer. Por ejemplo, se le puede indicar a una unidad de cinta magnética que se rebobine o avance un registro.
- **Estado**. Utilizadas para comprobar diversas condiciones de estado asociadas a un módulo de E/S y sus periféricos .
- **Transferencia**. Utilizadas para leer y/o escribir datos entre los registros del procesador y los dispositivos externos.

E/S DIRIGIDA POR INTERRUPCIONES

El problema de la E/S programada es que el procesador tiene que esperar mucho tiempo hasta que el módulo de E/S correspondiente esté listo para la recepción o la transmisión de más datos. El procesador, mientras está esperando, debe comprobar repetidamente el estado del módulo de E/S. Como re-
sultado, el nivel de rendimiento de todo el sistema se degrada gravemente.

Una alternativa es que el procesador genere un mandato de E/S para un módulo y, acto seguido, continúe realizando algún otro trabajo útil. El módulo de E/S interrumpirá más tarde al procesador para solicitar su servicio cuando esté listo para intercambiar datos con el mismo. El procesador ejecutará la transferencia de datos, como antes, y después reanudará el procesamiento previo.

Considere cómo funciona esta alternativa, primero desde el punto de vista del módulo de E/S. Para una operación de entrada, el módulo de E/S recibe un mandato de LECTURA del procesador. El módulo de E/S pasa entonces a leer los datos de un periférico asociado. Una vez que los datos están en el registro de datos del módulo, el módulo genera una interrupción al procesador a través de una línea de control. El módulo entonces espera hasta que el procesador pida sus datos. Cuando se hace la petición, el módulo sitúa sus datos en el bus de datos y ya está listo para otra operación de E/S.

Desde el punto de vista del procesador, las acciones correspondientes a una operación de lectura son las que se describen a continuación. El procesador genera un mandato de LECTURA. Salva el contexto (por ejemplo, el contador de programa y los registros del procesador) del programa actual y lo abandona, pasando a hacer otra cosa (por ejemplo, el procesador puede estar trabajando en varios programas diferentes a la vez). Al final de cada ciclo de instrucción, el procesador comprueba si hay interrupciones (Figura 1.7). Cuando se produce la interrupción del módulo de E/S, el procesador salva el contexto del programa que se está ejecutando actualmente y comienza a ejecutar un programa de manejo de interrupción que procesa la interrupción. En este caso, el procesador lee la palabra de datos del módulo de E/S y la almacena en memoria. A continuación, restaura el contexto del programa que había realizado el mandato de E/S (o de algún otro programa) y reanuda su ejecución.

Casi invariablemente, habrá múltiples módulos de E/S en un computador, por lo que se necesitan mecanismos para permitir que el procesador determine qué dispositivo causó la interrupción y para decidir, en caso de múltiples interrupciones, cuál debe manejar primero. En algunos sistemas, hay múltiples líneas de interrupción, de manera que cada módulo de E/S usa una línea diferente. Cada línea tendrá una prioridad diferente. Alternativamente, puede haber una única línea de interrupción, pero se utilizan líneas adicionales para guardar la dirección de un dispositivo. De nuevo, se le asignan diferentes prioridades a los distintos dispositivos.

Acceso directo a memoria

La E/S dirigida por interrupciones, aunque más eficiente que la E/S programada simple, todavía requiere la intervención activa del procesador para transferir datos entre la memoria y un módulo de E/S, ya que cualquier transferencia de datos debe atravesar un camino a través del procesador. Por tanto, ambas formas de E/S sufren dos inconvenientes inherentes:

1. La tasa de transferencia de E/S está limitada por la velocidad con la que el procesador puede comprobar el estado de un dispositivo y ofrecerle servicio.
2. El procesador está involucrado en la gestión de una transferencia de E/S; se deben ejecutar varias instrucciones por cada transferencia de E/S.

Cuando se van a transferir grandes volúmenes de datos, se requiere una técnica más eficiente: el acceso directo a memoria (Direct Memory Access, DMA). La función de DMA puede llevarla a cabo un módulo separado conectado en el bus del sistema o puede estar incluida en un módulo de E/S. En cualquier caso, la técnica funciona como se describe a continuación.

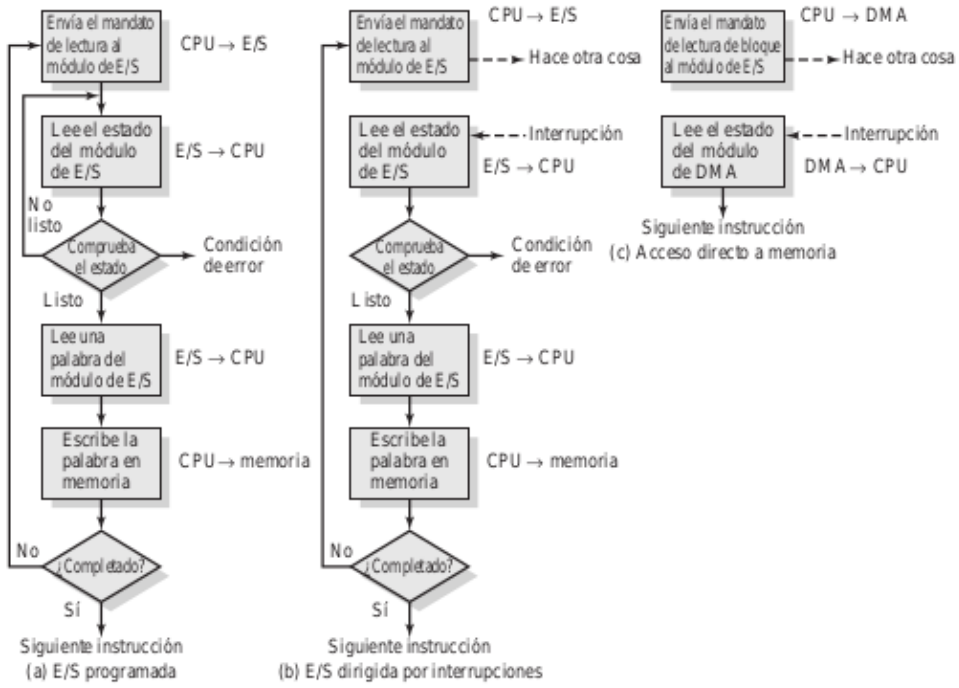
Cuando el procesador desea leer o escribir un bloque de datos, genera un mandato al módulo de DMA, enviándole la siguiente información:

- Si se trata de una lectura o de una escritura.
- La dirección del dispositivo de E/S involucrado.
- La posición inicial de memoria en la que se desea leer los datos o donde se quieren escribir.
- El número de palabras que se pretende leer o escribir.

A continuación, el procesador continúa con otro trabajo. Ha delegado esta operación de E/S al módulo de DMA, que se ocupará de la misma. El módulo de DMA transferirá el bloque completo de datos, palabra a palabra, hacia la memoria o desde ella sin pasar a través del procesador. Por tanto, el procesador solamente está involucrado al principio y al final de la transferencia.

El módulo de DMA necesita tomar el control del bus para transferir datos hacia la memoria o desde ella. Debido a esta competencia en el uso del bus, puede haber veces en las que el procesador necesita el bus y debe esperar al módulo de DMA. Nótese que esto no es una interrupción; el procesador no salva un contexto y pasa a hacer otra cosa. En su lugar, el procesador se detiene

durante un ciclo de bus (el tiempo que se tarda en transferir una palabra a través del bus). El efecto global es causar que el procesador ejecute más lentamente durante una transferencia de DMA en el caso de que el procesador requiera acceso al bus. Sin embargo, para una transferencia de E/S de múltiples palabras, el DMA es mucho más eficiente que la E/S dirigida por interrupciones o la programada.



Apéndice

Una técnica habitual para controlar la ejecución de llamadas a procedimiento y los retornos de los mismos es utilizar una pila. Este apéndice resume las propiedades básicas de las pilas y revisa su uso para el control de procedimientos.

Implementación de la pila

Una pila es un conjunto ordenado de elementos, tal que en cada momento solamente se puede acceder a uno de ellos (el más recientemente añadido). El punto de acceso se denomina cima de la pila El número de elementos de la pila, o longitud de la pila, es variable. Por esta razón, se conoce también a
una pila como una lista de apilamiento o lista donde el último que entra es el primero que sale (Last-In First-Out, LIFO).

La implementación de una pila requiere que haya un conjunto de posiciones dedicado a almacenar los elementos de la pila. En la Figura 1.25 se muestra una técnica habitual. Se reserva en memoria principal (o memoria virtual) un bloque contiguo de posiciones. La mayoría de las veces el bloque está parcialmente lleno con elementos de la pila y el resto está disponible para el crecimiento de la pila. Se necesitan tres direcciones para un funcionamiento adecuado, que habitualmente se almacenan en registros del procesador:

- **Puntero de pila**. Contiene la dirección de la cima de la pila. Si se añade un elemento (APILA) o se elimina (EXTRAE), el puntero se decrementa o se incrementa para contener la dirección de la nueva cima de la pila.
- **Base de la pila**. Contiene la dirección de la posición inferior en el bloque reservado. Se trata de la primera posición que se utiliza cuando se añade un elemento a una pila vacía. Si se hace un intento de extraer un elemento cuando la pila está vacía, se informa del error.
- **Límite de la pila**. Contiene la dirección del otro extremo, o cima, del bloque reservado. Si se hace un intento para apilar un elemento cuando la pila está llena, se indica el error.

Tradicionalmente, y en la mayoría de las máquinas actuales, la base de la pila está en el extremo con la dirección más alta del bloque de pila reservado, y el límite está en el extremo con la dirección más baja. Por tanto, la pila crece desde las direcciones más altas a las más bajas.

Llamadas y retornos de procedimientos

Una técnica habitual para gestionar las llamadas y los retornos de los procedimientos es utilizar una pila. Cuando el procesador ejecuta una llamada, se almacena (apila) la dirección de retorno en la pila. Cuando se ejecuta un retorno, se utiliza la dirección de la cima de la pila y se elimina (extrae) esa dirección de la pila. Es también necesario con frecuencia pasar parámetros en una llamada a procedimiento. Estos se podrían pasar en los registros. Otra posibilidad es almacenar los parámetros en la memoria justo después de las instrucciones de llamada. En este caso, el retorno debe estar en la posición siguiente a los
parámetros. Ambas técnicas tienen sus inconvenientes. Si se utilizan los registros, el programa llamado y el que realiza la llamada deben escribirse de manera que se asegure que los registros se utilizan apropiadamente. El almacenamiento de parámetros en memoria hace difícil intercambiar un número
variable de parámetros.

Una estrategia más flexible para el paso de parámetros es la pila. Cuando el procesador ejecuta una llamada, no sólo apila la dirección de retorno, sino también los parámetros que se desean pasar al procedimiento llamado. El procedimiento invocado puede acceder a los parámetros en la pila. Al retornar, los parámetros de retorno se pueden almacenar también en la pila, debajo de la dirección de retorno. El conjunto completo de parámetros, incluyendo la dirección de retorno, que se almacena en una invocación de procedimiento se denomina marco de pila.

Procedimientos reentrantes

Un concepto útil, particularmente en un sistema que da soporte a múltiples usuarios al mismo tiempo, es el de los procedimientos reentrantes. Un procedimiento reentrante es aquél en el que una única copia del código del programa se puede compartir por múltiples usuarios durante el mismo periodo de tiempo. El carácter reentrante de un procedimiento tiene dos aspectos fundamentales: el código del programa no puede modificarse a sí mismo y los datos locales de cada usuario deben almacenarse separadamente. Un procedimiento reentrante puede ser interrumpido e invocado por el programa que
causó la interrupción y, a pesar de ello, ejecutarse correctamente al retornar al procedimiento. En un sistema compartido, el carácter reentrante permite un uso más eficiente de la memoria principal. Se mantiene una copia del código del programa en memoria principal, pero más de una aplicación puede
llamar al procedimiento.

Por tanto, un procedimiento reentrante debe tener una parte permanente (las instrucciones que constituyen el procedimiento) y una parte temporal (un puntero de retorno al programa que realizó la llamada, así como espacio para las variables locales usadas por el programa). Cada instancia de la ejecución, llamada activación, de un procedimiento ejecutará el código en la parte permanente pero debe tener su propia copia de los parámetros y las variables locales. La parte temporal asociada con una activación en particular se denomina registro de activación.

La manera más conveniente de dar soporte a los procedimientos reentrantes es mediante una pila. Cuando se llama a un procedimiento reentrante, el registro de activación del procedimiento se puede almacenar en la pila. Por tanto, el registro de activación se convierte en parte del marco de pila que se
crea en la llamada a procedimiento.

2.- Estructura del Sistema Operativo

2.1. Objetivos y funciones de los sistemas operativos

Si un programador tuviera que desarrollar una aplicación como un conjunto de instrucciones en código máquina que se encargaran de controlar completamente el hardware del computador, se enfrentaría a una labor extremadamente compleja. Para facilitar esta tarea, se proporcionan un conjunto de programas de sistema. Algunos de estos programas se conocen como utilidades.

El programa de sistema más importante es el sistema operativo. El sistema operativo oculta los detalles del hardware al programador y le proporciona una interfaz apropiada para utilizar el sistema. Actúa como mediador, haciendo más fácil al programador y a la aplicación el acceso y uso de dichas utilidades y servicios. De forma resumida, el sistema operativo proporciona normalmente servicios en las siguientes áreas:

- **Desarrollo de programas**. El sistema operativo proporciona una variedad de utilidades y servicios, tales como editores y depuradores, para asistir al programador en la creación de los rogramas. Normalmente, estos servicios se ofrecen en la forma de utilidades que, aunque no forman parte del núcleo del sistema operativo, se ofrecen con dicho sistema y se conocen como herramientas de desarrollo de programas de aplicación.
- **Ejecución de programas**. Se necesita realizar una serie de pasos para ejecutar un programa. Las instrucciones y los datos se deben cargar en memoria principal. Los dispositivos de E/S y los ficheros se deben inicializar, y otros recursos deben prepararse.
- **Acceso a dispositivos de E/S**. Cada dispositivo de E/S requiere su propio conjunto peculiar de instrucciones o señales de control para cada operación.
- **Acceso controlado a los ficheros**. Para el acceso a los ficheros, el sistema operativo debe reflejar una comprensión detallada no sólo de la naturaleza del dispositivo de E/S (disco, cinta), sino también de la estructura de los datos contenidos en los ficheros del sistema de almacenamiento. Adicionalmente, en el caso de un sistema con múltiples usuarios, el sistema operativo puede proporcionar mecanismos de protección para controlar el acceso a los ficheros.
- **Acceso al sistema**. Para sistemas compartidos o públicos, el sistema operativo controla el acceso al sistema completo y a recursos del sistema específicos. La función de acceso debe proporcionar protección a los recursos y a los datos, evitando el uso no autorizado de los usuarios
  y resolviendo conflictos en el caso de conflicto de recursos.
- **Detección y respuesta a errores**. Se pueden dar gran variedad de errores durante la ejecución de un sistema de computación. Éstos incluyen errores de hardware internos y externos, tales como un error de memoria, o un fallo en un dispositivo; y diferentes errores software, tales como la división por cero o el intento de acceder a una posición de memoria prohibida. En cada caso, el sistema operativo debe proporcionar una respuesta que elimine la condición de error, suponiendo el menor impacto en las aplicaciones que están en ejecución. La respuesta puede oscilar entre finalizar el programa que causó el error hasta reintentar la operación o simplemente informar del error a la aplicación.
- **Contabilidad**. Un buen sistema operativo recogerá estadísticas de uso de los diferentes recursos y monitorizará parámetros de rendimiento tales como el tiempo de respuesta, información es útil para anticipar las necesidades de mejoras futuras y para optimizar el sistema. En un sistema multiusuario, esta información se puede utilizar para facturar a los diferentes usuarios.

El sistema operativo como gestor de recursos

Un computador es un conjunto de recursos que se utilizan para el transporte, almacenamiento y procesamiento de los datos, así como para llevar a cabo el control de estas funciones. El sistema operativo se encarga de gestionar estos recursos, pero es un sistema de control inusual en dos aspectos:

- Las funciones del sistema operativo actúan de la misma forma que el resto del software; es decir, se trata de un programa o conjunto de programas ejecutados por el procesador.
- El sistema operativo frecuentemente cede el control y depende del procesador para volver a retomarlo.

De hecho, el sistema operativo es un conjunto de programas. Como otros programas, proporciona instrucciones para el procesador. La principal diferencia radica en el objetivo del programa. El sistema operativo dirige al procesador en el uso de los otros recursos del sistema y en la temporización de
la ejecución de otros programas. No obstante, para que el procesador pueda realizar esto, el sistema operativo debe dejar paso a la ejecución de otros programas. Por tanto, el sistema operativo deja el control para que el procesador pueda realizar trabajo «útil» y de nuevo retoma el control para permitir
al procesador que realice la siguiente pieza de trabajo.

Una porción del sistema operativo se encuentra en la memoria principal. Esto incluye el kernel, o núcleo, que contiene las funciones del sistema operativo más frecuentemente utilizadas y, en cierto momento, otras porciones del sistema operativo actualmente en uso. El resto de la memoria principal contiene
programas y datos de usuario.

El procesador es también un recurso, y el sistema operativo debe determinar cuánto tiempo de procesador debe asignarse a la ejecución de un programa de usuario particular. En el caso de un sistema multiprocesador, esta decisión debe ser tomada por todos los procesadores.

Facilidad de evolución de un Sistema Operativo

Un sistema operativo importante debe evolucionar en el tiempo por las siguientes razones:

- Actualizaciones de hardware más nuevos tipos de hardware. Por ejemplo, las primeras versiones de los sistemas operativos UNIX e IBM OS/2 no empleaban un mecanismo de paginado porque ejecutaban en máquinas sin hardware de paginación.
- Nuevos servicios. En respuesta a la demanda del usuario o en respuesta a las necesidades de los gestores de sistema, el sistema operativo debe ofrecer nuevos servicios. Por ejemplo, si es difícil mantener un buen rendimiento con las herramientas existentes, se pueden añadir al sistema operativo nuevas herramientas de medida y control.
- Resolución de fallos. Cualquier sistema operativo tiene fallos. Estos fallos se descubren con el transcurso del tiempo y se resuelven. Por supuesto, esto implica la introducción de nuevos fallos.

La necesidad de cambiar regularmente un sistema operativo introduce ciertos requisitos en su diseño. Un hecho obvio es que el sistema debe tener un diseño modular, con interfaces entre los módulos claramente definidas, y que debe estar bien documentado. Para programas grandes, tal como el tí-
pico sistema operativo contemporáneo, llevar a cabo una modularización sencilla no es adecuado.

2.2. Principales logros

Los sistemas operativos se encuentran entre las piezas de software más complejas jamás desarrolladas. Esto refleja el reto de intentar resolver la dificultad de alcanzar determinados objetivos, algunas veces conflictivos, de conveniencia, eficiencia y capacidad de evolución. [DENN80a] propone cinco principales avances teóricos en el desarrollo de los sistemas operativos:

- Procesos.
- Gestión de memoria.
- Protección y seguridad de la información.
- Planificación y gestión de los recursos.
- Estructura del sistema.

3.- Arquitecturas monolíticas, microkernel y máquinas virtuales.

3.1. Micronúcleos

Un micronúcleo es la pequeña parte central de un sistema operativo que proporciona las bases para extensiones modulares. El enfoque de micronúcleo se popularizó por su uso en el sistema operativo Mach. En teoría este enfoque proporciona un alto grado de flexibilidad y modularidad.

Arquitectura micronúcleo

La filosofía existente en el micronúcleo es que solamente las funciones absolutamente esenciales del sistema operativo estén en el núcleo. Los servicios y aplicaciones menos esenciales se construyen sobre el micronúcleo y se ejecutan en modo usuario. Aunque la filosofía de qué hay dentro y qué hay fuera del micronúcleo varía de un diseño a otro, la característica general es que

muchos servicios que tradicionalmente habían formado parte del sistema operativo ahora son subsistemas externos que interactúan con el núcleo y entre ellos mismos; algunos ejemplos son: manejadores de dispositivos, servidores de archivos, gestores de memoria virtual, sistemas de ventana y servicios de seguridad.

La arquitectura del micronúcleo reemplaza la tradicional estructura vertical y estratificada en capas por una horizontal. Los componentes del sistema operativo externos al micronúcleo se implementan como servidores de procesos; interactúan entre ellos dos a dos, normalmente por paso de mensajes a través del micronúcleo. De esta forma, el micronúcleo funciona como un intercambiador de mensajes: válida mensajes, los pasa entre los componentes, y concede el acceso al hardware. El micronúcleo también realiza una función de protección; previene el paso de mensajes a no ser que el intercambio esté permitido.

Beneficios de una organización micronúcleo

- El micronúcleo impone una interfaz **uniforme** en las peticiones realizadas por un proceso. Los procesos no necesitan diferenciar entre servicios a nivel de núcleo y a nivel de usuario, porque todos los servicios se proporcionan a través de paso de mensajes.
- La arquitectura micronúcleo facilita la **extensibilidad**, permitiendo agregar nuevos servicios, así como la realización de múltiples servicios en la misma área funcional. Con la arquitectura micronúcleo, cuando se añade una nueva característica, sólo el servidor relacionado necesita modificarse o añadirse. Además, las modificaciones no requieren la construcción de un nuevo núcleo.
- Relacionado con la extensibilidad de una arquitectura micronúcleo está su **flexibilidad**. No sólo se pueden añadir nuevas características al sistema operativo, además las características existentes se pueden eliminar para realizar una implementación más pequeña y más eficiente.
- El monopolio que casi tiene Intel en muchos segmentos del mercado de la computación es improbable que dure indefinidamente. De esta forma, la **portabilidad** se convierte en una característica interesante en los sistemas operativos. En la arquitectura micronúcleo, todo o gran parte del código específico del procesador está en el micronúcleo. Por tanto, los cambios necesarios para transferir el sistema a un nuevo procesador son menores y tienden a estar unidos en grupos lógicos. Mayor sea el tamaño de un producto software, mayor es la dificultad de asegurar su fiabilidad.
  El programador del sistema tiene un número limitado de API que dominar y métodos limitados de interacción y, por consiguiente, es más difícil afectar negativamente a otros componentes del sistema.
- El micronúcleo nos lleva por sí mismo al **soporte de sistemas distribuidos**, incluyendo clusters controlados por sistemas operativos distribuidos. Cuando se envía un mensaje desde un cliente hasta un proceso servidor, el mensaje debe incluir un identificador del servicio pedido. Si se configura un sistema distribuido (por ejemplo, un cluster) de tal forma que todos los procesos y servicios tengan identificadores únicos, entonces habrá una sola imagen del sistema a nivel de micronúcleo. Un proceso puede enviar un mensaje sin saber en qué máquina reside el servicio pedido.
- Una arquitectura micronúcleo funciona bien en el contexto de un **sistema operativo orientado a objetos**. Un enfoque orientado a objetos puede servir para diseñar el micronúcleo y para desarrollar extensiones modulares para el sistema operativo. Los componentes son objetos con interfaces claramente definidas que pueden ser interconectadas para la realización de software a través de bloques de construcción. Toda la interacción entre componentes utiliza la interfaz del componente.

Diseño del micronúcleo

Debido a que los diferentes micronúcleos presentan una gran variedad de tamaños y funcionalidades, no se pueden facilitar reglas concernientes a qué funcionalidades deben darse por parte del micronúcleo y qué estructura debe implementarse.

El micronúcleo debe incluir aquellas funciones que dependen directamente del hardware y aquellas funciones necesarias para mantener a los servidores y aplicaciones operando en modo usuario. Estas funciones entran dentro de las categorías generales de gestión de memoria a bajo nivel, intercomunicación de procesos (IPC), y E/S y manejo de interrupciones.

Gestión de memoria a bajo nivel

El micronúcleo tiene que controlar el concepto hardware de espacio de direcciones para hacer posible la implementación de protección a nivel de proceso. Con tal de que el micronúcleo se responsabilice de la asignación de cada página virtual a un marco físico, la parte principal de gestión de memoria, incluyendo la protección del espacio de memoria entre procesos, el algoritmo de reemplazo de páginas y otra lógica de paginación, pueden implementarse fuera del núcleo. Por ejemplo, un módulo de memoria virtual fuera del micronúcleo decide cuándo traer una página a memoria y qué página presente en memoria debe reemplazarse; el micronúcleo proyecta estas referencias de página en direcciones físicas de memoria principal.

Esta técnica permite a un proceso fuera del núcleo proyectar archivos y bases de datos en espacios de direcciones de usuario sin llamar al núcleo. Las políticas de compartición de memoria específicas de la aplicación se pueden implementar fuera del núcleo.

Se recomienda un conjunto de tres operaciones de micronúcleo que pueden dar soporte a la paginación externa y a la gestión de memoria virtual:

- **Conceder (Grant)**. El propietario de un espacio de direcciones (un proceso) puede conceder alguna de sus páginas a otro proceso. El núcleo borra estas páginas del espacio de memoria del otorgante y se las asigna al proceso especificado.
- **Proyectar (Map)**. Un proceso puede proyectar cualquiera de sus páginas en el espacio de direcciones de otro proceso, de forma que ambos procesos tienen acceso a las páginas. Esto genera memoria compartida entre dos procesos. El núcleo mantiene la asignación de estas páginas al propietario inicial, pero proporciona una asociación que permite el acceso de otros procesos.
- **Limpiar (Flush)**. Un proceso puede reclamar cualquier página que fue concedida o asociada a otro proceso.

Para comenzar, el núcleo asigna toda la memoria física disponible como recursos a un proceso base del sistema. A medida que se crean los nuevos procesos, se pueden conceder o asociar algunas páginas del espacio de direcciones original a los nuevos procesos. Este esquema podría dar soporte a
múltiples esquemas de memoria virtual simultáneamente.

Comunicación entre procesos (Interprocess Communication)

La forma básica de comunicación entre dos procesos o hilos en un sistema operativo con micronúcleo son los mensajes.

Un **mensaje** incluye una cabecera que identifica a los procesos remitente y receptor y un cuerpo que contiene directamente los datos, un puntero a un bloque de datos, o alguna información de control del proceso. Normalmente podemos pensar que las IPC se fundamentan en puertos asociados a cada proceso. Un **puerto** es, en esencia, una cola de mensajes destinada a un proceso particular; un proceso puede tener múltiples puertos. Asociada a cada puerto existe una lista que indica qué procesos se pueden comunicar con éste. Las identidades y funcionalidades de cada puerto se mantienen en el núcleo.

Un proceso puede concederse nuevas funcionalidades mandando un mensaje al núcleo con las nuevas funcionalidades del puerto. Llegado a este punto sería conveniente realizar un comentario sobre el paso de mensajes. El paso de mensajes entre dos procesos que no tengan solapado el espacio de direcciones implica realizar una copia de memoria a memoria, y de esta forma está limitado por la velocidad de la memoria y no se incrementa con la velocidad del procesador. De esta forma, la investigación actual de los sistemas operativos muestra interés en IPC basado en hilos y esquemas de memoria compartida como la re-proyección de páginas —page remapping— (una sola página compartida por múltiples procesos).

Gestión de E/S e interrupciones

Con una arquitectura micronúcleo es posible manejar las interrupciones hardware como mensajes e incluir los puertos de E/S en los espacios de direcciones. El micronúcleo puede reconocer las interrupciones pero no las puede manejar. Más bien, genera un mensaje para el proceso a nivel de usuario que está actualmente asociado con esa interrupción. De esta forma, cuando se habilita una interrupción, se asigna un proceso de nivel de usuario a esa interrupción y el núcleo mantiene las

asociaciones. La transformación de las interrupciones en mensajes las debe realizar el micronúcleo, pero el micronúcleo no está relacionado con el manejo de interrupciones específico de los dispositivos.

Puede verse el hardware como un conjunto de hilos que tienen identificadores de hilo único y que mandan mensajes (únicamente con el identificador de hilo) a hilos asociados en el espacio de usuario. El hilo receptor determina si el mensaje proviene de una interrupción y determina la interrupción específica.

4.- Sistemas operativos de propósito específico.

4.1. SO de tiempo real

Características

Los sistemas operativos de tiempo real pueden ser caracterizados por tener requisitos únicos en cinco áreas generales:

- **Determinismo**. Realiza operaciones en instantes de tiempo fijos predeterminados o dentro de intervalos de tiempo predeterminados. Cuando múltiples procesos compiten por recursos y tiempo de procesador, ningún sistema puede ser totalmente determinista. En un sistema operativo de tiempo real, las solicitudes de servicio de los procesos son dirigidas por eventos externos y temporizaciones.
- **Reactividad**. Se preocupa de cuánto tiempo tarda el sistema operativo, después del reconocimiento, en servir la interrupción. La reactividad incluye los siguientes aspectos:
    - La cantidad de tiempo necesario para manejar inicialmente la interrupción y comenzar a ejecutar la rutina de servicio de la interrupción (RSI). Si la ejecución de la RSI necesita un cambio de proceso, entonces el retardo será mayor que si la RSI puede ser ejecutada dentro del contexto del proceso actual.
    - La cantidad de tiempo necesario para realizar la RSI. Esto depende, generalmente, de la plataforma hardware.
    - El efecto del anidamiento de interrupciones. Si una RSI puede ser interrumpida por la llegada de otra interrupción, entonces el servicio se retrasará.

  El determinismo y la reactividad juntos conforman el **tiempo de respuesta** a eventos externos.
- **Control del usuario**. En un sistema de tiempo real es esencial permitirle al usuario un control de grano fino sobre la prioridad de las tareas. El usuario debe ser capaz de distinguir entre tareas duras y suaves y de especificar prioridades relativas dentro de cada clase. Un sistema de tiempo real puede también permitirle al usuario especificar características como el uso de paginación en los procesos, qué procesos deben residir siempre en memoria principal, etc.
- **Fiabilidad**. es normalmente mucho más importante para los sistemas de tiempo real que para los que no lo son. un sistema de tiempo real ha de responder y controlar eventos en tiempo real. La pérdida o degradación de sus prestaciones puede tener consecuencias catastróficas: pérdidas económicas, daños en equipos importantes e incluso pérdida de vidas.
- **Operación de fallo suave**. es una característica que se refiere a la habilidad del sistema de fallar de tal manera que se preserve tanta capacidad y datos como sea posible. Un aspecto importante de la operación de fallo suave se conoce como estabilidad. Un sistema de tiempo real es estable si, en los casos en los que sea imposible cumplir los plazos de todas las tareas, el sistema cumplirá los plazos de sus tareas más críticas, de más alta prioridad, aunque los plazos de algunas tareas menos críticas no se satisfagan. Para cumplir los requisitos precedentes, los sistemas operativos de tiempo real incluyen de forma representativa características como el cambio rápido de proceso o hilo o la planificación expulsiva basada en prioridades.

Planificación de tiempo real

En un compendio de algoritmos de planificación de tiempo real, se observa que los distintos enfoques de la planificación dependen de cuando el sistema realiza análisis de planificabilidad; y si lo hace, de si se realiza estática o dinámicamente; y de si el resultado del análisis produce un plan de planificación de acuerdo al cual se desarrollarán las tareas en tiempo de ejecución. En base a estas consideraciones los autores identifican las siguientes clases de algoritmos:

- **Enfoques estáticos dirigidos por tabla**. En éstos se realiza un análisis estático de la factibilidad de la planificación. El resultado del análisis es una planificación que determina cuando, en tiempo de ejecución, debe comenzar a ejecutarse cada tarea.
- **Enfoques estáticos expulsivos dirigidos por prioridad**. También se realiza un análisis estático, pero no se obtiene una planificación. En cambio, el análisis se utiliza para asignar prioridades a las tareas, y así puede utilizarse un planificador expulsivo tradicional basado en prioridades.
- **Enfoques dinámicos basados en un plan**. La factibilidad se determina en tiempo de ejecución (dinámicamente) en vez de antes de comenzar la ejecución (estáticamente). Una nueva tarea será aceptada como ejecutable sólo si es posible satisfacer sus restricciones de tiempo. Uno de los resultados del análisis de factibilidad es un plan que se usará para decidir cuándo poner en marcha la tarea.
- **Enfoques dinámicos de mejor esfuerzo**. No se realiza análisis de factibilidad. El sistema intenta cumplir todos los plazos y aborta la ejecución de cualquier proceso cuyo plazo haya fallado.

Planificación por plazos

Generalmente, las aplicaciones de tiempo real no se preocupan tanto de la velocidad de ejecución como de completar (o comenzar)
sus tareas en los momentos más adecuados, ni muy pronto ni muy tarde, a pesar de la demanda dinámica de recursos u otros conflictos, sobrecarga de procesamiento y fallos hardware o software. Ha habido varias propuestas de enfoques más potentes y apropiados para la planificación de tareas de tiempo real. Todos ellos se basan en tener información adicional acerca de cada tarea. En su forma más general, puede utilizarse la siguiente información de cada tarea:

- **Tiempo de activación**. Momento en el cual la tarea pasa a estar lista para ejecutar. En el caso de una tarea repetitiva o periódica se tratará de una secuencia de tiempos conocida de antemano. En el caso de una tarea aperiódica, este tiempo puede ser conocido previamente, o el sistema operativo sólo podrá ser consciente de la tarea cuando ésta pase a estar lista.
- **Plazo de comienzo**. Momento en el cual la tarea debe comenzar.
- **Plazo de conclusión**. Momento para el cual la tarea debe estar completada. Las aplicaciones de tiempo real típicas tendrán plazos de comienzo o bien plazos de conclusión, pero no ambos.

4.2. Multiprocesamiento simétrico

A medida que ha evolucionado la tecnología de los computadores y el coste del hardware ha descendido, los diseñadores han visto cada vez más oportunidades para el paralelismo, normalmente para mejorar el rendimiento y, en algunos casos, para mejorar la fiabilidad.

Arquitectura SMP

Es útil ver donde encaja la arquitectura SMP (multiprocesamiento simétrico) dentro de las categorías de procesamiento paralelo. Flynn propone las siguientes categorías de sistemas de computadores:

- **Única instrucción, único flujo de datos (SISD) stream**. Un solo procesador ejecuta una única instrucción que opera sobre datos almacenados en una sola
memoria.
- **Única instrucción, múltiples flujos de datos (SIMD) stream**. Una única instrucción de máquina controla la ejecución simultánea de un número de elementos de proceso. Cada elemento de proceso tiene una memoria de datos asociada, de forma que cada instrucción se ejecuta en un conjunto de datos diferente a través de los diferentes procesadores. Los procesadores vectoriales y matriciales entran dentro de esta categoría.

- **Múltiples instrucciones, único flujo de datos (MISD) stream**. Se transmite una secuencia de datos a un conjunto de procesadores, cada uno de los cuales ejecuta una secuencia de instrucciones diferente. Esta estructura nunca se ha implementado.
- **Múltiples instrucciones, múltiples flujos de datos (MIMD) stream**. Un conjunto de procesadores ejecuta simultáneamente diferentes secuencias de instrucciones en diferentes conjuntos de datos. Los procesadores son de propósito general, porque deben ser capaces de procesar todas las instrucciones necesarias para realizar las transformaciones de datos apropiadas. MIMD se puede subdividir por la forma en que se comunican los procesadores. Si cada procesador tiene una memoria dedicada, cada elemento de proceso es en sí un computador. La comunicación entre los computadores se puede realizar a través de rutas prefijadas o bien a través de redes.

  Este sistema es conocido como un **cluster**, o multicomputador. Si los procesadores comparten una memoria común, entonces cada procesador accede a los programas y datos almacenados en la memoria compartida, y los procesadores se comunican entre sí a través de dicha memoria; este sistema se onoce como multiprocesador de memoria compartida.

Una clasificación general de los multiprocesadores de memoria compartida se basa en la forma de asignar procesos a los procesadores. Los dos enfoques fundamentales son maestro/esclavo y simétrico. Con la arquitectura maestro/esclavo, el núcleo del sistema operativo siempre ejecuta en un determinado procesador. El resto de los procesadores sólo podrán ejecutar programas de usuario y, a lo mejor, utilidades del sistema operativo. El maestro es responsable de la planificación de procesos e hilos. Una vez que un proceso/hilo está activado, si el esclavo necesita servicios (por ejemplo, una llamada de E/S), debe enviar una petición al maestro y esperar a que se realice el servicio.

En un multiprocesador simétrico (Symmetric Multiprocessor, SMP), el núcleo puede ejecutar en cualquier procesador, y normalmente cada procesador realiza su propia planificación del conjunto disponible de procesos e hilos. El núcleo puede construirse como múltiples procesos o múltiples hilos, permitiéndose la ejecución de partes del núcleo en paralelo. El enfoque SMP complica al sistema operativo, ya que debe asegurar que dos procesadores no seleccionan un mismo proceso y que no se pierde ningún proceso de la cola. Se deben emplear técnicas para resolver y sincronizar el uso de los recursos.

Organización SMP

En máquinas modernas, los procesadores suelen tener al menos un nivel de memoria cache, que es privada para el procesador. El uso de esta cache introduce nuevas consideraciones de diseño. Debido a que la cache local contiene la imagen de una porción de memoria principal, si se altera una palabra en una cache, se podría invalidar una palabra en el resto de las caches. Para prevenir esto, el resto de los procesadores deben ser alertados de que se ha llevado a cabo una actualización. Este problema se conoce como el problema de coherencia de caches y se suele solucionar con técnicas hardware más que con el sistema operativo.

Consideraciones de diseño de SO multiprocesador

Un sistema operativo SMP gestiona los procesadores y otros recursos del computador, de manera que el usuario puede ver al sistema de la misma forma que si fuera un sistema uniprocesador multiprogramado. Las principales claves de diseño incluyen las siguientes características:

- **Procesos o hilos simultáneos concurrentes**. Las rutinas del núcleo necesitan ser reentrantes para permitir que varios procesadores ejecuten el mismo código del núcleo simultáneamente.
- **Planificación**. La planificación se puede realizar por cualquier procesador, por lo que se deben evitar los conflictos. Si se utiliza multihilo a nivel de núcleo, existe la posibilidad de planificar múltiples hilos del mismo proceso simultáneamente en múltiples procesadores.

- **Sincronización**. Con múltiples procesos activos, que pueden acceder a espacios de direcciones compartidas o recursos compartidos de E/S, se debe tener cuidado en proporcionar una sincronización eficaz.

  Gestión de memoria. La gestión de memoria en un multiprocesador debe tratar con todos los aspectos encontrados en las máquinas uniprocesador, que se verán en la Parte Tres. Además, el sistema operativo necesita explotar el paralelismo hardware existente, como las memorias multipuerto, para lograr el mejor rendimiento. Los mecanismos de paginación de los diferentes procesadores deben estar coordinados para asegurar la consistencia cuando varios procesadores comparten una página o segmento y para decidir sobre el reemplazo de una página.
- **Fiabilidad y tolerancia a fallos**. El sistema operativo no se debe degradar en caso de fallo de un procesador. El planificador y otras partes del sistema operativo deben darse cuenta de la pérdida de un procesador y reestructurar las tablas de gestión apropiadamente.

4.3. SO distribuidos y de red

- **Sistema operativo de red**. En esta configuración hay una red de máquinas, normalmente estaciones de trabajo de un solo usuario, y una o más máquinas servidoras. Éstas proporcionan servicios de red o aplicaciones, tales como almacenamiento de ficheros y gestión de impresión. Cada computador tiene su propio sistema operativo. El sistema operativo de red es un añadido al sistema operativo local, que permite a las máquinas interactuar con los servidores. El usuario conoce la existencia de múltiples computadores y debe trabajar con ellos de forma explícita. Normalmente se utiliza una arquitectura de comunicaciones común para dar soporte a estas aplicaciones de red.
- **Sistema operativo distribuido**. Es un sistema operativo común compartido por una red de computadores. A los usuarios les parece un sistema operativo normal centralizado, pero les proporciona acceso transparente a los recursos de diversas máquinas. Un sistema operativo distribuido puede depender de una arquitectura de comunicaciones para las funciones básicas de comunicación, pero normalmente se incorporan un conjunto de funciones de comunicación más sencillas para proporcionar mayor eficiencia.

El intercambio de información entre computadores con finalidad cooperativa se conoce como comunicación de computadores. De forma similar, cuando uno o más computadores se interconectan a través de una red de comunicación, el conjunto de computadores se denomina red de computadores.
Ya que se requiere un nivel similar de cooperación entre un terminal y un computador, estos términos también se utilizan cuando alguna de las entidades de comunicación son terminales. En relación a la comunicación de computadores y redes de computadores, hay dos conceptos de suma importancia:

- Protocolos.
- Arquitectura de comunicaciones o arquitectura de protocolos.

Un **protocolo** se utiliza para comunicar entidades de diferentes sistemas. Los términos entidad y sistemas se utilizan en un sentido muy genérico. Algunos ejemplos de entidades son los programas de aplicación de usuario, paquetes de transferencia de ficheros, sistemas de gestión de bases de datos, servicios de correo electrónico y terminales. Algunos ejemplos de sistemas son computadores, terminales y sensores remotos. Fijarse que en algunos casos la entidad y el sistema en que reside son los mismos (por ejemplo, terminales). En general, una entidad es cualquier cosa capaz de enviar y recibir información, y un sistema es un objeto físico que contiene una o más entidades. Para que dos entidades se comuniquen con éxito, deben «hablar el mismo idioma». Lo que se comunica, cómo se comunica y cuándo se comunica, debe hacerse de acuerdo a unas convenciones entre las entidades involucradas. Las convenciones se denominan protocolos, que se pueden definir como un conjunto de reglas que gobiernan el intercambio de datos entre dos entidades. Los elementos principales de un protocolo son los siguientes:

- **Sintaxis**. Incluye cosas tales como formatos de datos y niveles de señales.
- **Semántica**. Incluye información de control para realizar coordinación y gestión de errores.
- **Temporización**. Incluye ajuste de velocidades y secuenciamiento.