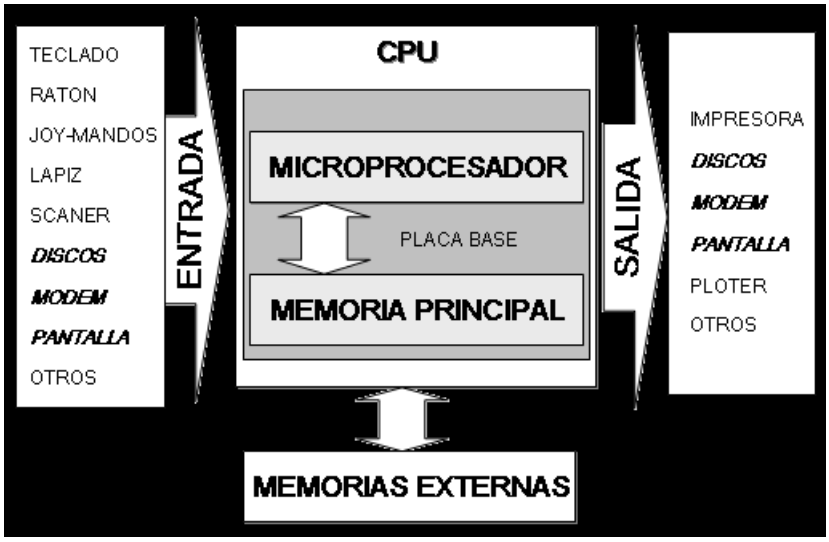
1.Dibujar un esquema básico de los componentes de un computador y sus interconexiones: CPU, memoria y dispositivos.



2. Diferencia entre interrupciones y excepciones ¿Cómo se gestionan?

Interrupción: señal que envía un dispositivo de E/S a la CPU para indicar que la operación de la que se estaba ocupando, ya ha terminado.

Tratamiento de las interrupciones:

-El manejador de interrupción procesa la interrupción, examinando la información de estado relacionada el evento que haya causado la interrupción.

-Cuando se completa el procesamiento de la interrupción, se recuperan los valores de los registros salvados en la pila y se restauran en los registros.

-Finalmente, se restauran de la pila los valores de la palabra de estado y del contador de programa, y se continúa con la ejecución del programa previamente interrumpido.

Excepción: una situación de error detectada por la CPU mientras ejecutaba una instrucción, que requiere tratamiento por parte del SO.

Tratamiento de las excepciones:

Cuando la CPU intenta ejecutar una instrucción incorrectamente construida, la unidad de control lanza una excepción para permitir al SO ejecutar el tratamiento adecuado. Al contrario que en una interrupción, la instrucción en curso es abortada. Las excepciones al igual que las interrupciones deben estar identificadas.

3. En una arquitectura de modo dual de funcionamiento ¿cómo obtenemos un servicio del SO?

Obtendremos el servicio del sistema operativo a través de una trampa, que es un tipo de interrupción síncrona que producirá una excepción, lo que a su vez genera un cambio de contexto a modo kernel, haciendo que se salve los registros PC y PSW, para después cargar en PC la dirección que hay en el vector de excepciones, para acceder a la rutina del sistema que queremos que se ejecute.

4. Diferencia entre entradas/salidas programadas y e/s gestionadas por interrupción.

La sincronización por programa (E/S programada) es la más sencilla de implementar en un computador, sin embargo, presenta algunos inconvenientes:

- Pérdida de tiempo: el computador no realiza trabajo útil en el bucle de espera

- Impide la realización de tareas periódicas, como la exigida por el refresco de una pantalla .

- Dificultades para atender varios periféricos

En la E/S programada el procesador tiene que esperar un tiempo considerable a que el módulo de E/S esté preparado para realizar la operación. El procesador espera comprobando repetidamente el estado del módulo de E/S, degradándose significativamente el rendimiento de la CPU. Para evitar este inconveniente se introdujo el sistema de interrupciones en los procesadores.

5.¿Cómo implementa un SO el concepto (abstracción) proceso?

Para implementar el modelo de procesos, el SO mantiene una tabla llamada tabla de procesos, con una entrada por proceso. Cada entrada contiene información acerca del estado del proceso, su contador de programa, apuntador a la pila …

6. ¿Qué es la imagen de un proceso?

Espacio en memoria para almacenar la 'pila de ejecución' (Estructura de tamaño intermedio, donde se almacenan datos temporales necesarios en un proceso.)

7. ¿A qué estado pasa un proceso que solicita un servicio del SO por el que posiblemente debe esperar? ¿cómo alcanza dicho estado?

Pasa al estado de preparado, es decir; el proceso está preparado para ejecutar cuando tenga oportunidad. El proceso alcanza dicho estado tras realizar una llamada al sistema.

8. ¿A qué estado(s) pasa el proceso que está actualmente ejecutandose cuando se produce una interrupción?

Puede pasar a estado bloqueado, donde el proceso está pendiente de un evento externo que le ha hecho bloquear.

También puede que se encuentre en estado de terminado, donde el proceso realiza una llamada al sistema solicitando su terminación.

9. Los sistemas operativos actuales se construyen para que los procesos cooperen en la multiprogramación. ¿Cómo se hace para que cooperen?

El sistema operativo selecciona uno de los trabajos y empieza su ejecución. Eventualmente, el trabajo tiene que esperar por algo. En un sistema no multiprogramado el CPU debe esperar si hacer nada. En un sistema con multiprogramación, el sistema operativo simplemente seleccionará otro trabajo y lo ejecutará. Cuando ese trabajo necesite esperar, el CPU será asignado a otro trabajo y de esta forma continuará. Eventualmente el primer trabajo habrá terminado su espera y obtendrá el CPU nuevamente. De esta manera, sí siempre existe algún trabajo para ejecutar, el CPU nunca estará ocioso.