Sistema operativos: es el componente de software principal que se compone de un conjunto de programas.

Software: el código ejecutable, los programas y datos que se van a utilizar.

Tipos de software:

- usuario (seria la interface)
- aplicación
- sistema operativo (sería el encargado de hacer todas las gestiones)

computadora: es una unidad compuesta por un CPU con una memoria principal y una conexión de entrada y salida de periféricos (discos también pueden ser periféricos)

CPU: este está almacenado en el microprocesador

MP(memoria principal): en él está el sistema operativo el cual es la interface y el encargado de hacer las tareas, tiene una constante comunicación con el disco para hacer las tareas de las aplicaciones

Disco tiene la funcionalidad de tener una permanencia después del apagado

La CPU se divide en dos partes, las unidad de control UC y la unidad aritmético lógica ALU

CP contador del programa tiene la siguiente extracción a programar (es el que indica donde está la instrucción)

|CO (código de operación)- DIR |(seria la ubicación en la que se hace la tarea)

Bus S: es el encargado de direccionar las direcciones

Bus de datos o de memoria (bus M):

RS: registros

Bit de unidades elementales para representar un valor binario ue se pueden representar con 0 y 1.

La ALU tiene un registro acumulados (AC) en la alu se hacen todo tipo de operaciones aritméticas.

ERS entrada de registro de selecciones

Salida de registro de instrucciones

Los valores están siempre en le hexadecimal

El contador de programa tiene direcciones de programas la dirección contendrá una instrucción no un dato, a parte el sistema puede actualizar el contador con una línea de código.

Tipos de direccionamiento:
Directo: es la manera que se obtiene la dirección efectiva.

La dirección efectiva es la información que se tiene en la memoria principal.

Indirecto: el dato que esta dentro de la dirección no será un dato, sino que será un dato que vuelva a buscar en la memoria para buscar la dirección efectiva.

CO|DR|DIR = M en el indirecto M debe contener el mismo tamaño del dir para poder con la palabra que vendrá de la memoria.

Lo que esta en el DIR es la dirección efectiva

La ventaja de esto sobre la directa es que puede utilizar mas memoria sacrificando señalar las memoria dos veces

2^b= canti de combinaciones posibles con “b” bits

Inmediato o dirección lateral

En el inmediato lo que este en la campo dir es el dato, se utiliza esto cuando se reuiqere algo de forma especifica

Direccionamiento relativo

¿Qué es?
El direccionamiento relativo se utiliza principalmente en instrucciones de **salto condicional o incondicional**, donde el destino **no es una dirección fija**, sino que se calcula **sumando un desplazamiento (offset)** al contador de programa actual (**PC**).

¿Por qué se usa?
Es útil cuando el código puede cambiar de ubicación en memoria (por ejemplo, si el programa es cargado en distintos lugares), ya que el salto es relativo al punto actual, no a una dirección absoluta.

Indexado: las direcciones se pueden incrementar o decrementar

¿Qué es?
El direccionamiento indexado se usa cuando queremos acceder a **datos organizados secuencialmente en memoria**, como **arreglos o vectores**. Implica una **dirección base** (puede ser un registro) y un **índice/desplazamiento** (también en un registro o constante).

¿Por qué se usa?
Porque permite recorrer estructuras de datos fácilmente sin tener que escribir una instrucción diferente para cada elemento.

Memoria de 16GB cib oakavras de 64 bits el registro de instrucción tiene 32 bits campo de dirección 16 bits

Directo 2^{16}

Indirecto $2^{32*2} = 2^{64}$ porque pasa dos veces en la memoria la palabra

C=0 ninguno porque nunca se hace una dirección (no se puede referencia direcciona memoria en el inmediato)

28/03

Sistema operaitvo

memorias: conjunto de bits que la memoria utiliza para armar instrucciones

punto de memoria

FF= flip flop

Todo lo que este echo por semiconductores y por ejemplo se va la luz, la información que estaba almacenada se pierde (memoria volatil)

Tiempo de suceso: tiempo de lectura

Caudal: es el tiempo para sacar la memoria

tipos de memoria:
la cache se puede definir en la memoria principal (pero todas las memorias tienen cache)

Principal: memoria RAM memroia volátil y accesos directo

Secundaria: disco rigido, hdd,sdd memoria no volátil

La diferencia entre la rigida y la sdd es la rapidez al acceso de la información

Videovigilancia(para no perder la esperanza de vida de la memoria)

NASS o servudores trabajan en sincronía con los diferentes discos

Memoria fuera de línea o terseria?

Estas son para ser transportados no volátil

La micro SD o prendrive de los celulares se consideran memoria secundaria en los celulares

Jerarquia

1 registro de cpu

2 cache cpu es para del registro

3memoria principal

4 memoria secundaria

5 alamacenamiento fuera de línea

Los de arriba tienen:

- mayor costo por bit
- mayor velovidad
- cada ves mas arriba esta mas cerca del cpu

la cache cachea información de niveles inferiores

memoria de alta velicidad que almacena infroamcion reciente para accederla lo antes posible

todos los perifericos son los que realizan la interaccion de entrada y salida

el programa o cpu es el inicia una interaccion de entrada y salida

DMA darleal periférico acceso o un banco de memoria

DMA (Direct Memory Access – Acceso Directo a Memoria)

✅ ¿Qué es?

Es una técnica que permite que **los periféricos accedan directamente a la memoria RAM**, **sin la intervención del procesador (CPU)** para transferir datos.

🎁 ¿Para qué sirve?

Aumenta el rendimiento del sistema al liberar al procesador de tareas repetitivas de transferencia de datos, como:

- Leer datos de un disco duro o tarjeta SD
- Transferir audio desde un micrófono a la memoria
- Enviar una imagen desde la memoria a la pantalla

🔄 ¿Cómo funciona?

1. El CPU configura el **controlador DMA** con:
    - Dirección de origen (ej: periférico o memoria)
    - Dirección de destino
    - Cantidad de datos
2. El **controlador DMA** toma el control del bus de datos.
3. Realiza la transferencia **directamente entre periférico y RAM**.
4. Al terminar, el DMA puede generar una **interrupción** para avisar a la CPU.

📌 Ejemplo:

Un periférico de audio envía datos a la RAM por DMA. El CPU sigue ejecutando instrucciones sin ocuparse de copiar byte por byte.

Modelo de north bridy

Modelo de North Bridge (Puente Norte)

✅ ¿Qué es?

Es parte de una arquitectura clásica de placas madre (motherboards) de computadoras, en la que el sistema está dividido en dos chips principales:

- **North Bridge (Puente Norte):** maneja la **alta velocidad** (CPU, RAM, GPU)
- **South Bridge (Puente Sur):** maneja la **baja velocidad** (USB, disco, audio)

Este modelo ya está casi obsoleto, pero es clave para entender cómo funcionaban los sistemas antes de la integración moderna (como en los procesadores actuales tipo Intel i-series o AMD Ryzen).

🔄 ¿Qué hacía el North Bridge?

- Conectaba la **CPU** con la **RAM (memoria principal)**
- Comunicaba la CPU con la **tarjeta gráfica (GPU)** en AGP o PCIe
- A veces también controlaba el **bus de expansión rápido**

SO clase 2:
* todo lo que se conecta a la computadora son recursos de la computadora esto se dice para dar una interfaz amigable al usuario

*El so es la parte del programa que siempre este corriendo cubriendo los puntos anteriores (administrar recursos y dar una interfaz amigable)

Las utilidades pueden ser las bibliotecas

Servicio que proporciona el SO

*contabilidad(métricas de usuo de programas, de como van)

*el bucle de instrccion siempre va a donde dice el contador de programa

*el SO sede el control para hacer cierta acción hasta que ña terminación y luego vuelve a tener el poder

Planificacion y gestión de recursos (equitatividad)

Precesa(sinónimo de programa

Equitatividad

Respuesta difencial: discernir entre clases de trabajo con diferentes requerimientos

Eficiencia: maximizar la productividad, minimizando el tiempo de respuesta

Composicion del SO:
depende de cada SO, se divide en 3 partes KErnel(nucleo), programas del sistema, programa de aplicación

Varia normalmente según las necesidades del consumidor final

Kernel: tiene la función de vincular con el hardware(es la ultima palabra a la hora de tomar desiciones)

Tipos de kernel?
monolítica(incluye la mayor cantidad de funciones en el código principal)

Microkernel (mantener el kernel con la menor cantidad de funciones o servicio posibles, en ete caso cada acción va a tener que llamar al sitema (ahorra miemoria para hacer llamadas))

Hibrida(la combinación de estas dos anteriores)

Procesos: programa en ejecución, una intrancia de un programa ejecutado en un computador, entroduce que se puede asignar y ejecutar en un procesador unidad de actividades que se caracteriza por la ejecución de instrucciones
tiene dos elementos esenciales:

*código de programa

´conjunto de datos asociados

PCB(bloque de control de proceso) es una estructura de datos, que conteien la infromacion de da proceso

Identificada, priridad, contador de programa, puntero a memoria, datos de contexto, información de estado de e/s, información de auditoria

Según cada SO estos datos varian

*Estados(nuevos, pendientes, listo, floqueado, etc)

*PRiodiad cuanto mas bajo el numero mayor la prioridad

Hilo parte de un proceso, este surve para que partes de un mismo proceso que se pueden pueden ejecutar de forma paralela.

Estado de proceso: estos moderan cada proceso a la hora de analizarlo

Modelo de los datos:

Estrada-> no ejecuta ->(activación)<-(detencion) ejecuta ->salida

6to estado: aparece el estado de suspendio el cual sirve para reservar memoria
nivel de planificación:

- Largo plazo encargado de llenar la memoria
- Mediano es ek que trabajo con la memoria virtual (equilibra la población de la memoria)
- Corto planificador del cpu es el mas rápido nano y micro sd
- e/s planificación

Algoritmo de planificación:
Orientado al usuario tiempo de respuesta

Oriendado al sistema (criterio en el rendimiento o perfimance del sistema

Creiterio general

*uso de CPU (cuanto lo usa)

*Rendimiento

Tiempo de retorno, de espera y respuesta

Medidas cuantitativas

w=tiempo en el que estuvo en todo el sistema

e= cuanto tiempo estuvo ejecutándose el proceso

s= tiempo total requetido por el proceso

Modo decisión:
sin expulsión: una vez arranco un proceso este se ejecuta hasta que se termina o se suspende

Con expulsión: púede volver a la cola de listo en el momento de ejecución

RounRobin: el si q es muy chi hay muchos procesos, se consumen muha cpu y a que hay muchos cambios de contexto

TEP tiempo de espera promedio

TRP tiempo de respuesta promedio

HRRN: **HRRN** significa **Highest Response Ratio Next** (*Siguiente con la mayor razón de respuesta*).
Es un **algoritmo de planificación de procesos** (scheduling) usado en sistemas operativos para decidir **qué proceso ejecutar después** en la CPU.

Objetivo de HRRN:

Equilibrar entre:

- **Justicia**: evitar que procesos cortos esperen mucho.
- **Eficiencia**: mejorar el tiempo de respuesta sin ser tan injusto como SJF (Shortest Job First).

Cambio de contexto: normalmente vale una unidades de tiempo seria el cambio que hace la cpu al cambiar el analises de un proceso a otro

TR(tiempo de respuesta) desde el proceso entro al sistema hasta que termino de ejecutarse

Tiempo de retorno el tiempo que demoro desde que entro al sistema hasta que se fue

Respuesta (Respues-arribo)

Retorno (Fin- Arribo)

Espera(Retorno-Ejecucion)

Spn o sjf(eligen el menor tiempo de servicio y lo ejecuta completo(primero el proceso mas lento))

Srt o srtf (menor tiempo restante) elegie el que menor tiempo de serivcio retante le queda

* Cuando se va agregando complejidad este genera que agrega tiempo de cómputo y consume memoria de la CPU

*inanición procesos que nunca se ejecutan

*retroalimentación feedback son métodos con expulsión

*seria una decontruccion ya que se hacen varios procesos de diferentes formas

Hilo: no es un proceso, parte internas de un mismo proceso que se pueden hacer de forma independiente

Subproceso: es otro proceso invocado por una mayor jerarquía

*si los hilos pertenecen a un mismo proceso estos pueden ser independientes entre si

Hilos (mejor repsuesta, escabilidad, económico, Mejor uso de CPU Facilita la organización de tareas concurrentes**,** Comunicación rápida entre hilos

- 🔁 **Mejor tiempo de respuesta**
    - Al compartir memoria y recursos, **cambiar de un hilo a otro es más rápido** que entre procesos.
    - Ideal para tareas que requieren alta interactividad (ej: interfaces gráficas, servidores web).
- ⚖️ **Escalabilidad**
    - En sistemas **multinúcleo**, múltiples hilos pueden correr en **paralelo real**, aprovechando al máximo el hardware.

- **💰 Menor costo (económicos en recursos)**
    - Crear un hilo consume **menos memoria y tiempo** que crear un proceso.
    - Compartir el mismo espacio de direcciones evita duplicar recursos.
- **🔄 Mejor uso de CPU (paralelismo y concurrencia)**
    - Mientras un hilo espera (por ejemplo, por E/S), otro puede ejecutarse.
    - Aumenta la **eficiencia general del sistema**.
- **□ Facilita la organización de tareas concurrentes**
    - Permite dividir una tarea compleja en **subtareas independientes** que pueden ejecutarse en paralelo (por ejemplo, un navegador: un hilo para la interfaz, otro para la red, otro para la renderización).
- **📡 Comunicación rápida entre hilos**
    - Como comparten memoria, la **comunicación entre hilos es más rápida** que entre procesos (no requiere pipes, sockets, etc.).

Sepera los proceso de carga de los elementos de presentación de una aplicacionde las consulta y validación de datos (mejor respuesta y uso de los recursos)

Unidad de ejecuccion que se realiza dentro un proceso, donde puede ver otras unidades de ejecuccion

Hilos =! Procesos

Multihilo monoprocesados

- Mejor estructura de programa
- Solapamiento de e/s
- Baja penalización en el cambio de contexto entre hilos, normalmente al kernel

Multiprocesado seira la suma entre el monoprocesado y el paralelismo se llama paralelismo real.

Osea varios hilos a nivel de usuario a un hilo de kernel, si un heli de estos se bloquea se bloqueara toda

*modelo uno a un hilo de usuario un hilo de kernel, lo bueno, mayor paralelismo, lo malo que reduce la -performance de la aplicación

Mucho a muchos varios hilos de usuario y varios kernel user>kernel hilos

Mayor dificular de implmentacion

Dos nivel cambio la idea de muchos a muchos pero da la posibilidad de uno a uno

Mejor redactado lo de arriba:
Multihilo en sistemas monoprocesadores

- Permite una **mejor estructuración del programa**, facilitando la organización y la modularidad del código.
- Favorece el **solapamiento de operaciones de entrada/salida (E/S)**, ya que mientras un hilo espera, otro puede ejecutarse.
- Presenta una **baja penalización en los cambios de contexto entre hilos**, ya que generalmente se mantienen dentro del mismo espacio de memoria y son gestionados por el kernel.

□ Multiprocesamiento

- Es la combinación de la ejecución multihilo con múltiples núcleos o procesadores.
- Se conoce como **paralelismo real**, ya que varios hilos pueden ejecutarse **físicamente en paralelo**, cada uno en un núcleo distinto.

□ Modelos de hilos

1. Modelo "Muchos a uno" (Many-to-One)

- **Varios hilos de usuario** se asignan a **un solo hilo de kernel**.
- Ventajas:
  - Bajo costo de gestión.
- Desventajas:
  - Si **un hilo se bloquea**, **todos los demás se bloquean** también.
  - **No hay verdadero paralelismo** en sistemas multiprocesador, ya que solo se ejecuta un hilo del grupo a la vez.

2. Modelo "Uno a uno" (One-to-One)

- Cada **hilo de usuario** se asigna a **un hilo del kernel**.
- Ventajas:
  - **Mayor paralelismo**, ya que múltiples hilos pueden ejecutarse simultáneamente en diferentes núcleos.
- Desventajas:
  - **Mayor sobrecarga** para el sistema operativo, ya que debe gestionar más hilos a nivel del kernel.
  - Puede **reducir el rendimiento** si se crean demasiados hilos, debido a la saturación del sistema.

Modelo de hilos de dos niveles (Two-Level Model)

- Combina las características de los modelos **"muchos a muchos" (many-to-many)** y **"uno a uno" (one-to-one)**.
- Permite que:
    - **Varios hilos de usuario** se asignen a **menos o igual número de hilos del kernel** (como en many-to-many).
    - Pero también se puede crear una **asignación directa de uno a uno** para ciertos hilos (como en one-to-one), si el sistema lo requiere.

✅ Ventajas:

- **Flexibilidad**: se pueden ejecutar múltiples hilos de usuario en menos hilos de kernel, optimizando recursos.
- **Paralelismo real**: al permitir también hilos uno a uno, se puede aprovechar el multiprocesamiento real.
- **Mejor control de rendimiento**: el sistema puede elegir qué hilos merecen una asignación directa al kernel (por ejemplo, los que son críticos).

❌ Desventajas:

- **Mayor complejidad** en la implementación del sistema operativo.
- Requiere una **gestión más sofisticada** de las asignaciones entre hilos de usuario y del kernel.

}

Concurrencia: conceptos clave: sección cirtica, interbloqueo, curculo cicioso, exclusión mutua, condición de carrera, inanición

Condicion de carrera: sucede cuando multiples procesos o hilos leen y escriben datos de manera que el resultado final depende del orden de ejecución de las instrucciones en los multiples procesos.

Sección critica: sección de un código dentro de un proceso que hace uso datos compartidos con otra, la idea es que mientras un proceso corre en su sección critica, otro no lo haga para garantizar la consistencia de datos

requisitos para una buena solución

- No pueden ver dos procesos de forma simultanea dentro de sis refiones
- No puede hacer suposiciones acerca de las velicidad o numero de cpu
- Ningún proceso que se ejecuta fuera de una región critica puede bloquear otra
- Ningún proceso puede esperar para siempre entrar a su sección critica

Soluciones:
*desactivar instrucciones

*variable cerrada
*alternancia estricta(como un while)
*algoritmo de perterson (es una forma de estudiar el problema, no memorizar hay que entenderlo)
*dormir y despertar (el proceso se va a dormir hasta que lo llamen,)wakeup y sleep

*espera ocupada: maximicar la eficiencia de la cpu

*exclusión mutua: dos procesos están en su sección critica y se excluyen mutuamente
wake up sleep: puede perderse la señal de despertar, se puede arreglar con el bit de espera

*semáforo y monitoreos

Semáforo: al momento que se señalice el proceso no se puede interrumpir el proceso hasta que termine, en caso de multiples CPU se usa protección de hardware
variables semáforo
llenas: contabiliza las ranuras llevas
vacia contabiliza las ranuras vacías
mutex: binaria es como variable cerrada (true o false)

Si el semáforo no puede hacer la primicia en un proceso es porque esta vacio (o bloqueada) o es que hay un proceso en curso

El problema es que si colocamos mal el orden de los semáforos puedo bloquear los procesos primero vacia y luego mutex

Monitores: nivel de abstracción mas alto que usa los semaforros pero no nos hacemos cargo sino que el compilador lo hace, asi no nos hacemos cargos de orden básicamente usa doci y despide los métodos wait y signal

Monitores: nivel de abstracción más alto que usa los semáforos, pero no nos hacemos cargo directamente, sino que el compilador lo hace. Así no nos hacemos cargo del orden. Básicamente, usa **wait** y **signal** como métodos.

Deadlock (interbloqueo) es la situación en la que **dos procesos se bloquean** esperando que se liberen **recursos** mutuamente, sin poder continuar.

Condiciones necesarias para que ocurra un deadlock:

- **Exclusión mutua**
- **Retención y espera**
- **No apropiación**
- **Espera circular**

Soluciones:

- **Ignorarlo** (algoritmo del avestruz)
- **Detección y recuperación** (por ejemplo, un sistema de reloj con *timeouts* que invoque un proceso de inspección)

- **Evitarlos dinámicamente**, asignando cuidadosamente los recursos (un sistema de bloqueo similar al algoritmo del banquero)
- **Prevención**, eliminando alguna de las cuatro condiciones necesarias