

Estructura y funciones

Thursday, 6 February 2020 6:30 PM

Sistema Von Neumann



- | Pueden Tienen funciones
- | Registro → Tienen nombre
- | Memoria → Tiene dirección
- | * register int i;
- | //En lenguaje C

- Registros son determinados por el compilador
 - por humano en bajo nivel
- Registros más rápidos que memoria

ADD Ax, Bx, Cx 2.6 Hz → $\frac{1}{2}$ [ns] //Registros (Procesador)

ADD [Ax], [Bx], [Cx] //Acceso de memoria ≈ 100 [ns]

↑ latencia

6Gb - ancho de banda

- ∴ Solución
- ↳ Caché
- | - Memoria oculta
 - | - Tiene 3 niveles
 - | - Lo gestiona el hardware
 - Memory Manage Unit (MMU)

Ejemplo de acumuladores: Intel 8086/8088

| Registros de propósito general | | |
|-------------------------------------|----|------------------------------------|
| AH | AL | AX (Acumulador) |
| BH | BL | BX (Base) |
| CH | CL | CX (Contador) |
| DH | DL | DX (Datos) |
| Registros índices | | |
| SI | | Source Index (Índice origen) |
| DI | | Destination Index (Índice Destino) |
| BP | | Base Pointer (Puntero Base) |
| SP | | Stack Pointer (Puntero de Pila) |
| Registro de Bandera | | Flags (Banderas) |
| CF | OF | Overflow Flag |
| PF | SF | Parity Flag |
| ZF | DF | Zero Flag |
| AF | IF | Adjust Flag |
| Registros de Segmentos | | |
| CS | | Code Segment (Segmento de Código) |
| DS | | Data Segment (Segmento de Datos) |
| ES | | ExtraSegment (Segmento Extra) |
| SS | | Stack Segment (Segmento de Pila) |
| Registro apuntador de instrucciones | | Instruction Pointer |
| IP | | |

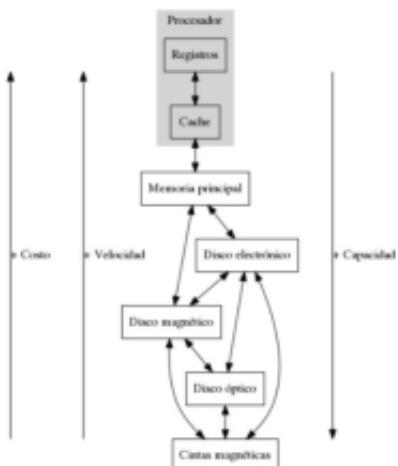
Arquitectura

Registros **bandera** (vector de booleanos): Overflow, Dirección, Interrupción, Trampa/depuración, Signo, Cero, Acarreo auxiliar, Paridad, Acarreo



Figura: Imagen de Wikipedia

- **Procesadores RISC:** A partir de los 1980
- Planteamiento base de instrucciones *sencillas y regulares*
 - Fáciles de codificar en un procesador pequeño (en número de transistores)
 - La arquitectura **RISC** más conocida hoy: **ARM**
- Tipicamente ≥ 32 registros *largos* (32, 64bits) de propósito general
 - Mas algunos de propósito específico



- Tiempo de acceso
↳ Latencia
- Tasa de transferencia
↳ Que velocidad sostiene

Cuadro: Velocidad y gestor de los principales niveles de memoria.
(Silberschatz, Galvin, Gagne; p.28)

| Nivel | 1 | 2 | 3 | 4 |
|----------------|-------------------|----------------|----------------|------------|
| Nombre | Registros | Cache | Memoria princ. | Disco |
| Tamaño | <1KB | <16MB | <64GB | >100GB |
| Tecnología | Multipuerto, CMOS | SRAM CMOS | CMOS DRAM | Magnética |
| Acceso (ns) | 0.25-0.5 | 0.5-25 | 80-250 | 5,000,000 |
| Transf (MB/s) | 20,000-100,000 | 5,000-10,000 | 1,000-5,000 | 20-150 |
| Administra | Compilador | Hardware | Sist. Op. | Sist. Op. |
| Respaldoado en | Cache | Memoria princ. | Disco | CD o cinta |

Ejemplo



Gunnar Wolf

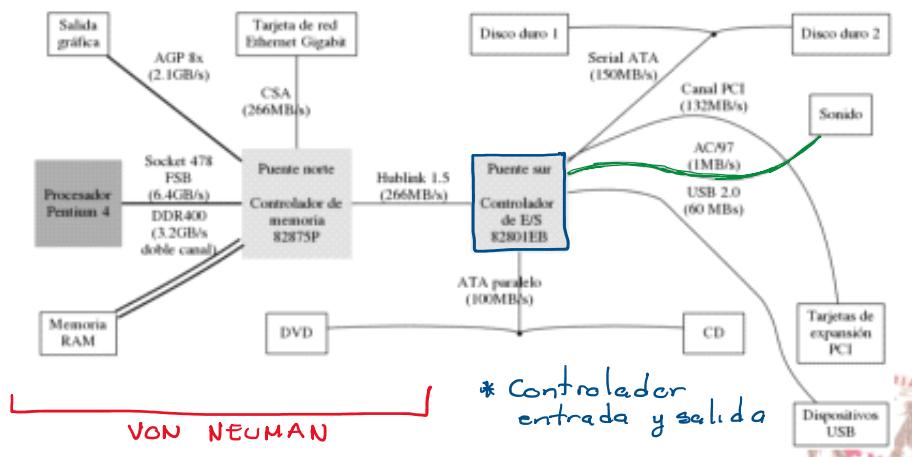
Estructuras básicas
Fronteras del CPU
Conectando hacia afuera
Interrupciones y excepciones
Multiprocesamiento

Almacenamiento primario y secundario

- El procesador sólo puede manejar directamente a la memoria principal
 - Se le conoce también como **almacenamiento primario**
 - Sólo éste es parte de lo que la arquitectura von Neumann llama **computadora**
- **Discos, cintas, almacenamiento estado sólido** son **almacenamiento secundario**
 - Todas las computadoras lo manejan a través de **controladores**

► Arquitectura

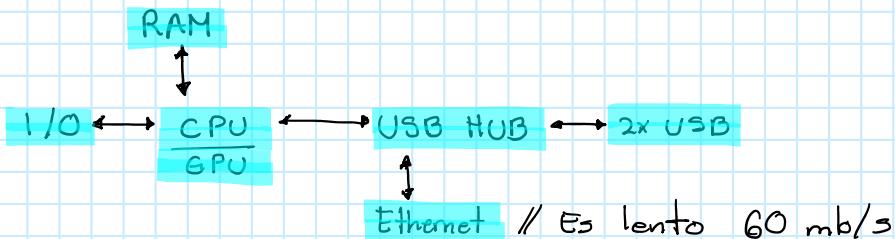
→ Intel 875 (2003)



- * Memoria para datos
 - * Memoria para instrucciones
 - * Saturación de canales → CONTENCIÓN
- ∴ Se dedicó uno de SONIDO por UDB

for eso es
una mezcla
entre ambas
arquitecturas

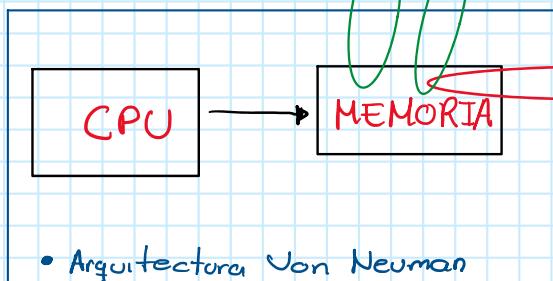
► Raspberry Pi (Arquitectura)



Tipos

Tuesday, 11 February 2020

5:47 PM

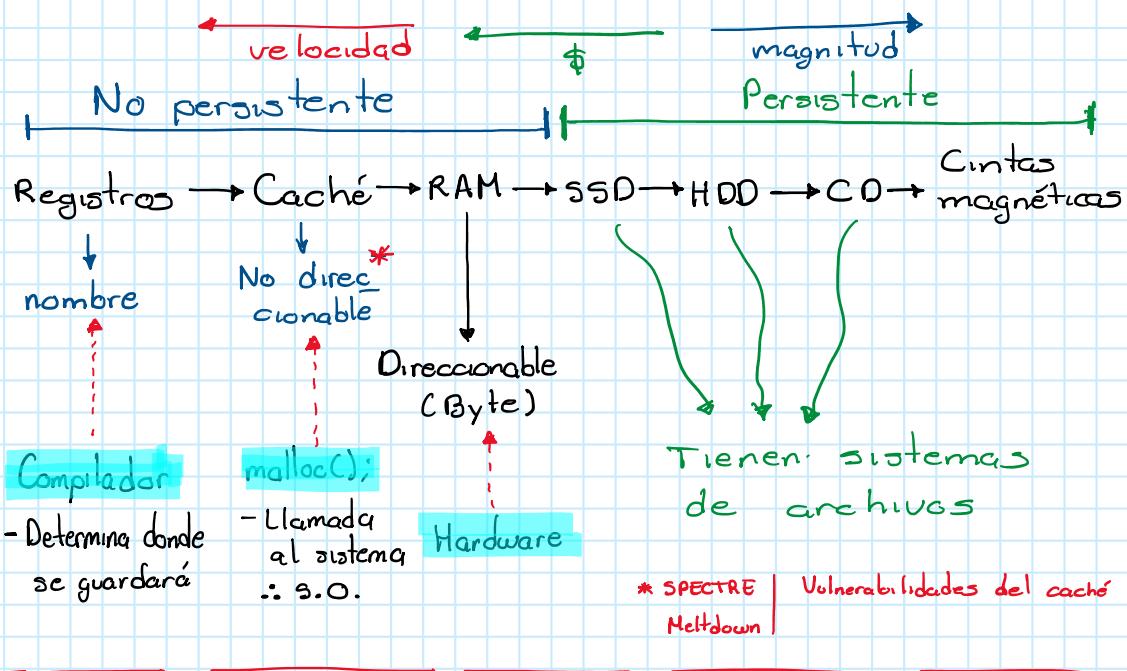


► Arquitectura

Subsistema Entrada/salida
03fov

* La puerta para salir es el sistema operativo

► Jerarquía de Almacenamiento



moto monitor
protegido ≠ privilegiado

Privilegios en hardware

murryfly
root/administrador

Privilegios en Sistema Operativo

Privilegios
↳ PC

Sistema Operativo
MONOLÍTICO

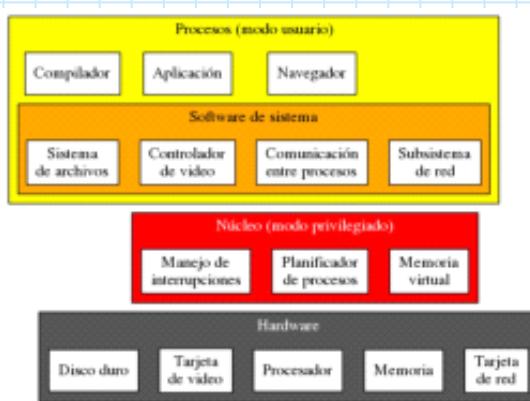
Sistema Operativo
MicroKernel

- Un sólo espacio de memoria
- Todo el código SO al mismo nivel
- MÁS FÁCIL de diseñar
- MÁS DIFÍCIL de depurar, debe considerarse cualquier cambio

→ Linux al inicio
(Proyecto de Torvalds)



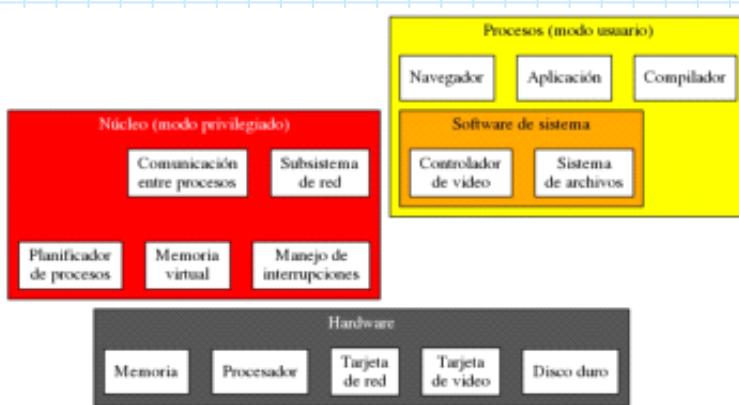
- Sistema Operativo pequeño
 - Muchos procesos especializados a nivel sistema
 - Muy modular
 - Pero determinar el diseño es complejo
 - Interfaces claras y limpias
- Windows en su rediseño de los 90



NOTA

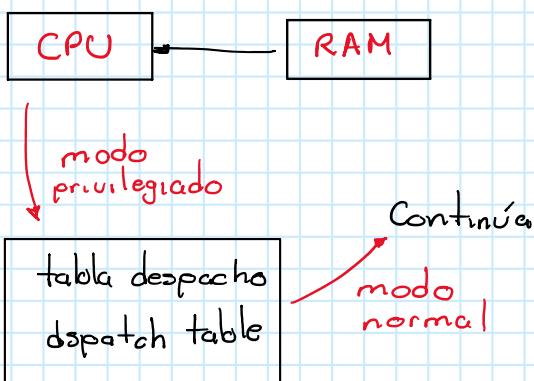
Siempre un sistema monolítico será más rápido que una microKernel

* La realidad de los sistemas actuales => Híbridos



► Interrupción y Excepciones

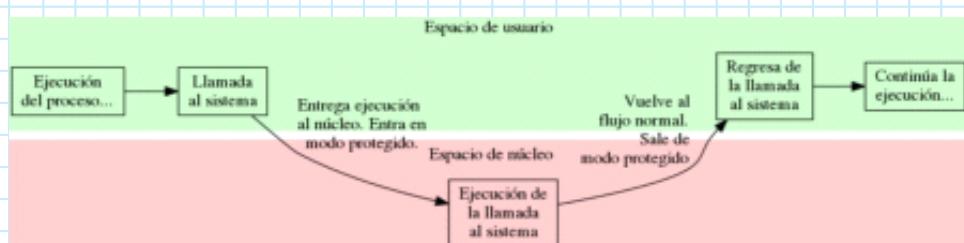
► Interrupción y Excepciones



- * Salta en modo protegido o privilegio para checar tabla de despacho

Llamada al sistema → printf(" ")

↑ Para hacer esto literalmente hago llamado a una excepción y por lo tanto llamamos al sistema operativo (Ej decir la dispatch table)



// Llamadas al sistema

Acciones que necesitan del S.O.

(Llamadas al sistema)

- ① Control de procesos
- ② Manipulación → Archivos
→ Dispositivos
- ③ Mantenimiento de información
- ④ Comunicaciones

- Puedo envolver al API del sistema y permite mostrar la traza de ejecución.

\V7ZELF //Ejecutables linux

strace _____ //Muestra todas las llamadas al S.O.
pwd

→ Acceso directo a memoria (DMA)

Problema: En la sección de un proceso en qué está limitado por entrada-salida

∴ Transferencia de información se vuelve un cuello de botella

∴ DMA → Orientado a dispositivos de gran ancho de banda

- CD - RED - Memoria y caché

↳ Transferencia en bloques empleando un controlador

- Dirección física base de memoria

- Cantidad de datos a transferir

* Contención de base de memoria

Tipos

Thursday, 13 February 2020 5:45 PM

//Resumen

Interrupción

- ① CPU salta a modo privilegiado
- ② Buscar en tabla de despacho
- ③ Busco volver al modo normal

Excepción

- ① Tropiezo interno
 - irregular
 - emular
 - corregir
- ② Puede determinar al proceso a continuar

Llamadas al sistema

- ① Invocado por el programa (Fingir tropiezo)

►DMA (Direct Memory Access)

Programmed I/O

```
while len(buffer) > 0  
    parte = buffer[0...16]  
    buffer = buffer[17...-1]  
    write(ALGUN_DISP, parte)
```

DMA

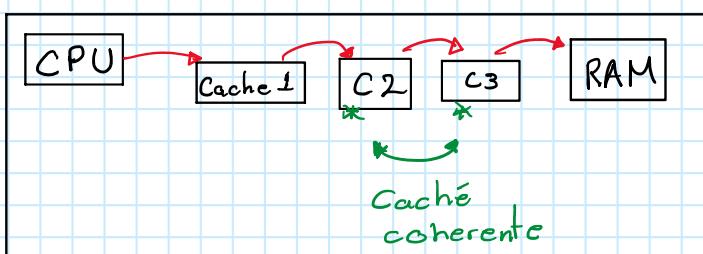
```
start-transfer(buffer,  
len(buffer), OUT, ALGUN_DISP)
```

Dir inicio

Longitud dirección Puerto dispositivo

* Viene desde el SPOOL

//Caché Coherente



* Es mucho más caro.

→ Multiprocesamiento

- Cuando tenemos muchos procesadores

→ Multi programación

- Muchos procesos

Multitarea

Cuando tienes muchos procesadores

Muchos procesos

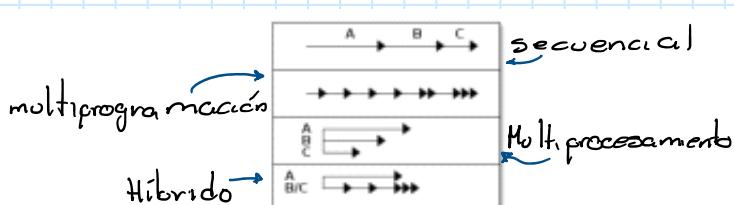


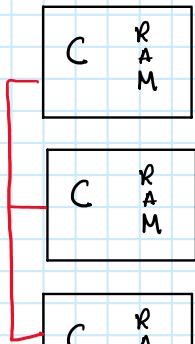
Figura: Esquema de la ejecución de tres procesos en un sistema secuencial, multiprogramado, multiprocesado, e híbrido

- Sincronización
 - ↳ DOMAR a la concurrencia
- Parallelización

- El CPU es más rápido que el acceso a memoria
 - ∴ Nació → los "núcleos" o cantidad de procesadores
 - ↳ 2005 // No todos los S.O soportaban multiprocesadores
- SMP \triangleq Multiprocesamiento simétrico, todos los procesadores son iguales
- Multiprocesamiento Asimétrico
 - Procesadores con distinta arquitectura
 - ↳ Se dedican a una tarea específica
 - Caso de tarjetas gráficas (GPU)

► NUMA (Non-Uniform Memory Access)

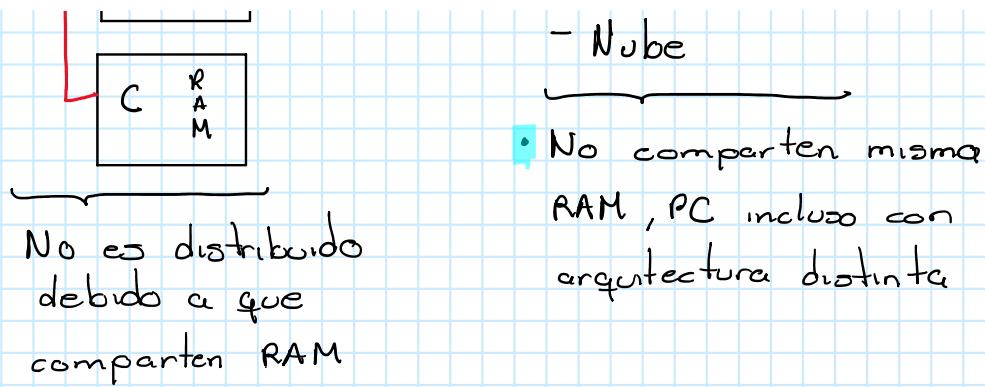
- Procesadores de la misma arquitectura
- Concepto "Memoria Cercana" y "Memoria lejana"



COMPUTO DISTRIBUIDO

- Clusters
- Grids
- Nube





Cúmulos (clusters)

- Computadoras conectadas por una red local de alta velocidad
- Cada una corre su propia instancia de sistema operativo y programas
- Principales orientaciones:
 - Alto rendimiento Cómputo matemático, cálculos...
 - Alta disponibilidad Prestación de servicios críticos
 - Balanceo de cargas Atención a solicitudes complejas, que pueden saturar a servidores individuales
- Típicamente son equipos homogéneos y dedicados
- Muy comunes en universidades; bajo costo, altas prestaciones



Mallas (grids)

- Computadoras distribuidas geográficamente
- Conectadas sobre Internet (o redes de área amplia)
- Pueden ser heterogéneas (en capacidades y en arquitectura)
- Presentan *elasticidad* para permitir conexiones/desconexiones de nodos en el tiempo de vida de un cálculo

Cómputo en la nube

- Caso específico y *de moda*
- Partición de recursos (*cliente-servidor*)
- Orientado a la *tercerización* de servicios específicos
- La *implementación* de un servicio deja de ser relevante → Procesos opacos
- Conceptos relacionados:
 - Servicios Web
 - Plataforma como servicio (*PaaS*)
 - Software como servicio (*SaaS*)
 - Infraestructura como servicio (*IaaS*)



- Software como servicio (*SaaS*)
- Infraestructura como servicio (*IaaS*)

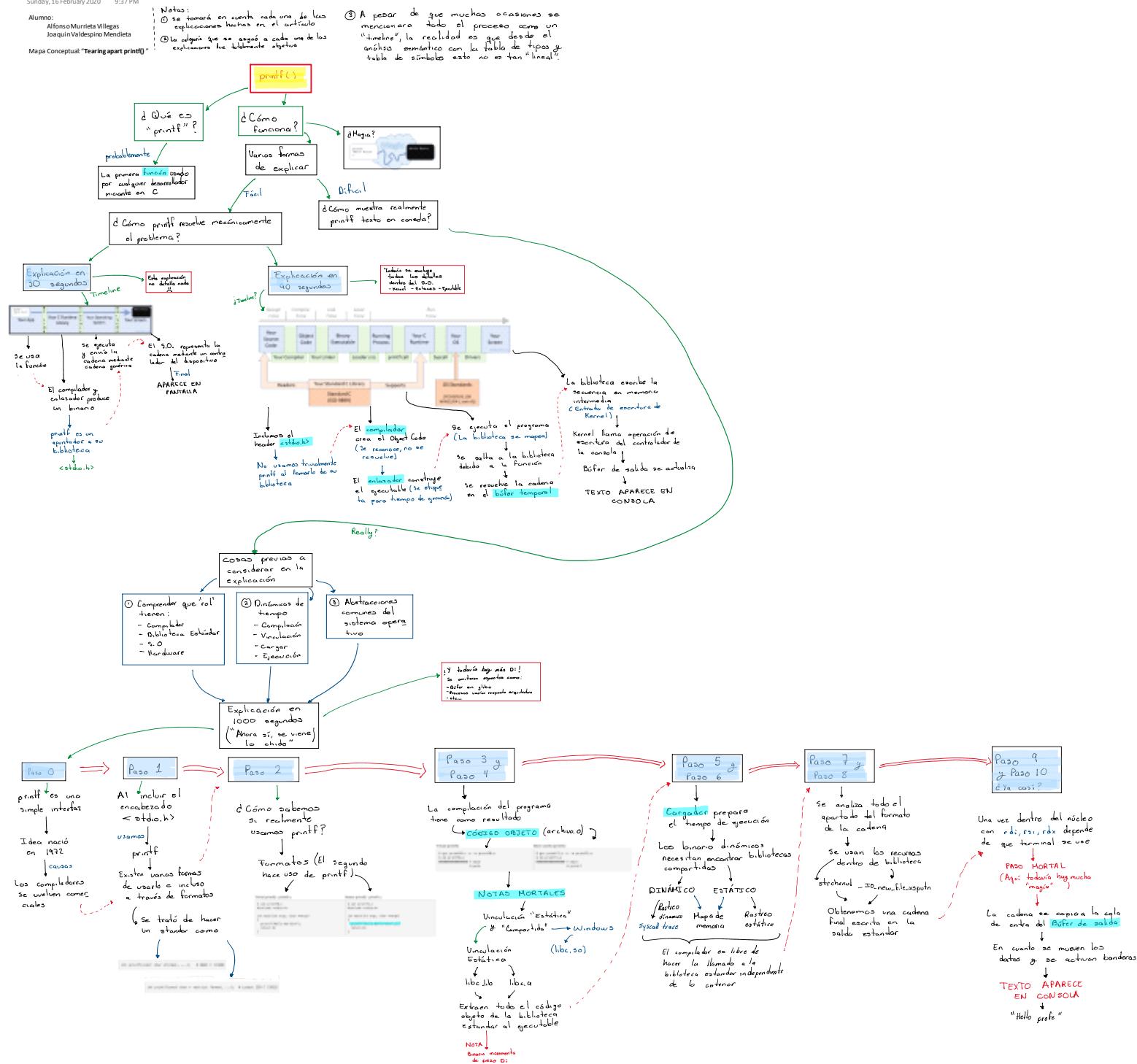


PROYECTO 1 - Mapa Conceptual

Sunday, 16 February 2020 9:37 PM

Alumno:
Alfonso Murrieta Villegas
Joaquín Valdespino Mendieta

Mapa Conceptual "Tearing apart printf"



Tipos y Paralelismo

Tuesday, 18 February 2020 5:43 PM

SWP

(Multiprocesamiento)
Simétrico

- Todos los procesadores son iguales
- Necesidad de coherencia
 - ↳ Acceso a misma memoria

AWP

(Multiprocesamiento)
asimétrico

- Distinta arquitectura
- Coprocesadores o procesadores adyacentes

big.LITTLE → Como se destina "demanda" o "consumo energético" a los procesadores

NUMA

- Non Uniform Memory Access
 - ↳ Los procesadores pueden ver toda la memoria, sin embargo, va variando la velocidad respecto a la lejanía del CPU.

Multiprocesamiento

- SWP
- AWP
- NUMA

Computo Distribuido

- Cluster / Grid / Malla
- Muchas computadoras
- Red alta velocidad
- Cada nodo tiene su S.O.

Computo en la nube

- Caso específico y de moda
- Partición de recursos (cliente-servidor)
- Orientado a la tercerización de servicios específicos

Cómputo en la nube

- Caso específico y *de moda*
- Partición de recursos (*cliente-servidor*)
- Orientado a la *tercerización* de servicios específicos
- La *implementación* de un servicio deja de ser relevante → Procesos opacos
- Conceptos relacionados:
 - Servicios Web
 - Plataforma como servicio (*PaaS*)
 - Software como servicio (*SaaS*)
 - Infraestructura como servicio (*IaaS*)



Límites del paralelismo

- Gene Amdahl (1967) observó que la ganancia derivada de parallelizar el código llega a un límite natural: El tiempo requerido por la *porción secuencial* del código
- La *porción paralelizable* puede repartirse entre varios procesadores, pero siempre hay una importante proporción no paralelizable
 - Inicialización
 - Puntos de control/acumulación de datos
 - Interacción con el usuario
 - etc.



Ley de Amdahl

$T(1)$ Tiempo de ejecución del programa con un sólo procesador

$T(P)$ Tiempo de ejecución con P procesadores

t_s Tiempo requerido por la porción secuencial

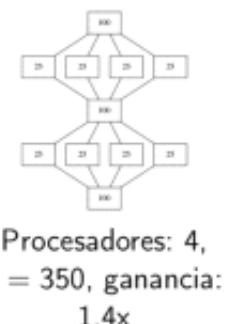
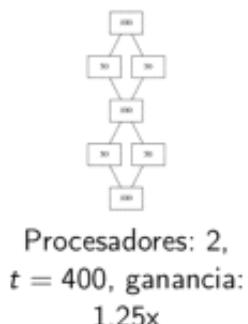
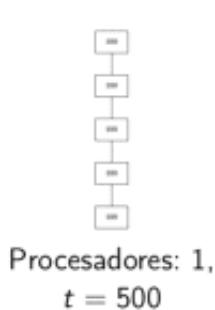
$t_p(P)$ Tiempo requerido para la porción paralela entre P procesadores

$$g = \frac{T(1)}{T(P)} = \frac{t_s + t_p(1)}{t_s + \frac{t_p(1)}{P}}$$



Estructuras básicas
Fronteras del CPU
Conectando hacia afuera
Interrupciones y excepciones
Multiprocesamiento

Ilustrando la ley de Amdahl (1)



Ley de Amdahl: ejecución de un programa con 500 unidades de tiempo total de trabajo con uno, dos y cuatro procesadores.



Ilustrando la ley de Amdahl (2)

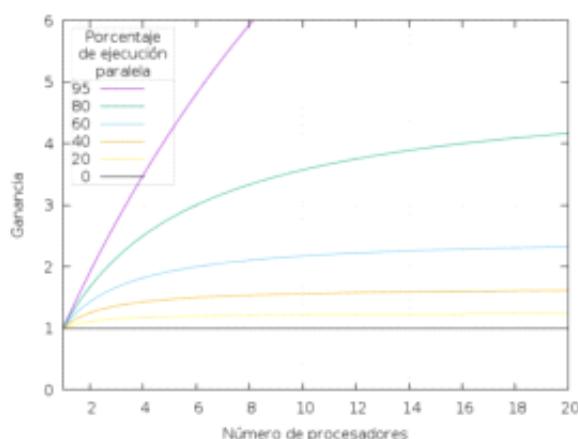


Figura: Ganancia máxima al parallelizar un programa, según la Ley de Amdahl



Estructuras básicas
Fronteras del CPU
Conectando hacia afuera
Interrupciones y excepciones
Multiprocesamiento

Desmenuzando a Amdahl

- El paralelismo puede aumentar fuertemente nuestro

- El paralelismo puede aumentar fuertemente nuestro rendimiento
 - ... ¿O no?
- Incluso con un 95 % de ejecución paralela, se llega a un techo de ganancia al llegar a los 20 CPUs
 - Cada procesador tiene un costo económico no trivial
 - ¿Vale realmente la pena ir migrando hacia un paralelismo cada vez mayor?



- d Entonces por qué renació?
 - Surgieron (Resurgieron) problemas de gran tamaño

- Ley Amdahl disminuyó el interés del público por los algoritmos en paralelo.
- En 1988 John Gustafson obtuvo ganancias de 120x en supercomputadoras de 1024 procesadores.