

ESTRUCTURA DE COMPUTADORES

impartido por: Manuel García Vázquez en la E.U.I.T.I.O. (<http://www.euitio.uniovi.es>)
redactado por: Miguel Herrero Obeso (<http://www.miguelherrero.cjb.net>)
actualizado el: 01/06/2004

Nota: Estos apuntes pretenden ser un reflejo de lo explicado en clase y están pensados para compaginarlos con las transparencias de la asignatura. No me responsabilizo de los suspensos que originen estos apuntes. Si consideras que alguna parte debería ser corregida, hazmelo saber enviándome un correo-e a través de mi página web.

Introducción a los sistemas operativos multitarea

Sistema operativo = Gestor de software
Hardware + S.O. = Máquina virtual

Multitarea:

Cada determinado tiempo se sustituye la tarea que ejecuta el procesador por otra. El sistema operativo elige que tarea va a ocupar la CPU.

¿Cuándo se utiliza el sistema operativo?

Cuando se produce una llamada al sistema (acceder a API)
Se produce una interrupción (el periférico avisa al S.O. de que tiene información)
Se produce una excepción (tipo de interrupción que genera el procesador)

Cuando el sistema operativo toma el control no implica necesariamente el cambio de tarea. Las causas por las que el sistema operativo toma el control, se pueden anidar. La nomenclatura utilizada depende de la arquitectura del procesador. Es necesario disponer de un temporizador que asegure una interrupción al menos cada T tiempo para que el sistema operativo tome el control.

Llamada a los servicios del S.O.

Se produce cuando una aplicación necesita acceder al hardware. En Intel, las instrucciones utilizadas son syscall (llama al sistema operativo a través de su API) y sysret (instrucción de retorno, es decir, que el sistema operativo devuelve el control a la aplicación).

Interrupción: mecanismo hardware por el que un periférico notifica al procesador que tiene información que comunicar. El procesador puede aceptar o rechazar la interrupción, para ejecutar la rutina de interrupción o no. Por último se ejecuta sysret.

Excepción: se pasa el control al sistema operativo cuando se produce una situación anormal durante la ejecución de la instrucción. Es el propio procesador el que provoca esta interrupción especial (cuando encuentra, por ejemplo, una división por cero).

La instrucción Sysret tiene tres alternativas:

- Que se devuelva la ejecución a una instrucción por delante de la que generó la excepción.
- Que se devuelva la ejecución a la instrucción que generó la excepción.
- Que no se pueda retornar (pantallazo azul en Windows).

La CPU

Soporte a los sistemas operativos multitarea:

Vamos a implementar un pequeño sistema operativo sobre el computador teórico siguiendo los siguientes pasos:

1. Se ejecuta una tarea y se produce una interrupción del temporizador.
2. Los registros SR y PC se almacenan en la pila de la tarea. Se transfiere el control al sistema operativo.
3. El sistema operativo guarda el resto de los registros de la CPU.
4. Se conmuta la pila a la del sistema operativo (se sustituye R7).
5. Se ejecuta la rutina de planificación.
6. Retorno del proceso de planificación con el valor de la próxima tarea a ejecutar.
7. Se restaura el estado de dicha tarea. Se recuperan sus registros y se escriben en la CPU.
8. Se retorna a la tarea a ejecutar y se recupera el SR y el PC.
9. La tarea esta ejecutándose.

El algoritmo anterior tiene algunas deficiencias:

- El sistema operativo se basa en un temporizador manual.
- Una tarea corriente podría desactivar la recepción de interrupciones y apropiarse de la CPU (mediante la instrucción CLI).
- Una tarea puede escribir en las direcciones de memoria del sistema operativo y modificarlo a su antojo, es decir, puede acceder a posiciones de memoria privilegiadas.
- Cada proceso guarda el PC y el SR. Esto debería ser almacenado por el sistema operativo.
- La CPU teórica no soporta excepciones.

Algunas posibles soluciones a estas deficiencias son:

- Marcar las instrucciones especiales (como CLI) reservadas para el sistema operativo. Se conoce como establece niveles de privilegio y se utiliza añadiendo un bit en el registro de estado.
- Marcamos zonas de memoria privilegiadas, es decir, reservadas exclusivamente para el S.O.
- La protección entre zonas de memoria de distintas tareas se gestiona a través de la memoria en sí.
- Utilizar una pila en el operativo que almacene los registros de cada tarea (incluyendo el PC y el SR).
- Introducir manejadores de excepciones con el mismo mecanismo que las interrupciones.

Mejoras de rendimiento

Cada instrucción se ejecuta en pasos o ciclos de reloj (medidos en Hertzios). Para mejorar el rendimiento podemos aumentar la frecuencia de reloj de la CPU, es decir, ejecutar mas pasos en el mismo tiempo.

Ejemplo:

```
COMP R0, R1 = 5 ciclos de reloj
```

```
Con 1KHz = 200 instrucciones por segundo.
```

```
Con 1MHz = 200000 instrucciones por segundo.
```

```
Con 1GHz = 200000000 instrucciones por segundo.
```

¿Cómo se consigue mejorar la frecuencia?

Reduciendo la separación entre las puertas lógicas que componen la CPU.

Podemos hacer otra mejora para aumentar el rendimiento reduciendo el número de pasos o ciclos por instrucción. Esto se consigue mediante la **segmentación en la ejecución de instrucciones**.

Mediante la segmentación se busca que cada parte del procesador se encuentre el máximo de tiempo ocupada. Se aprovecha la ejecución para buscar la siguiente instrucción y viceversa.

Con este sistema, en general, aumentaremos notablemente el rendimiento. Como desventajas podemos destacar que se incrementa la complejidad de la unidad de control y que el incremento máximo de velocidad solo es teórico (puede haber segmentos 'sueños' que se ejecuten de forma aislada).

Replicación de unidades funcionales

Replicamos los recursos internos del procesador (varias ALUs, varios buses internos, varias unidades de control, etc).

De esta forma se aprovecha mejor la segmentación, pero introduce mayor complejidad en el sistema.

Computador superescalar: aquél que puede trabajar simultáneamente sobre varias instrucciones.

Computador multiprocesador: aquél cuyo procesador está replicado.

La arquitectura IA-32 (Intel con arquitectura de 32bits)

Introducción histórica

En 1978 sale al mercado el procesador **8086/8088**.

IBM decidió hacer computadores 'baratos' para la gente (PC) y empleó el procesador citado anteriormente. El computador incluía el sistema operativo DOS.

IBM dejó la arquitectura del computador abierta, es decir, cualquiera podía saber como estaba construido internamente.

El 8086/8088 tenía 16 bits en cuanto a los registros de la CPU, pero el bus de direcciones tenía 20 bits.

El 8088 era una versión reducida y tenía un bus de datos de 8 bits, mientras que el 8086 tenía un bus de datos de 16 bits.

Con los 20 bits del bus de direcciones se podían direccionar $2^{20} = 1\text{Mb}$. Este mega se dividía en 640 Kb para la memoria y el resto para interfaces de entrada/salida.

Acceso a memoria mediante segmentos

Intel divide la memoria en segmentos y crea un **registro de segmento** que indica el tamaño de los segmentos. Las direcciones se dividen en segmento + desplazamiento.

Más tarde se lanza el **80186/80188**, versión del anterior que a pesar de incluir alguna novedad, no tuvo mucho éxito.

En 1982 se lanza el **80286**. Este procesador establece un **modo protegido** (para permitir multitarea), pero para mantener la compatibilidad con el 8086, incluye un **modo real**. Además se utilizan 24 bits en el bus de direcciones, lo que permite direccionar 16Mb. En modo protegido se podía utilizar lo que se denomina **memoria virtual**.

Para pasar de un modo a otro había que resetear la máquina, con lo que la mayoría de los usuarios utilizaban simplemente el modo real. Cuando se utilizaba este modo, solo se permitía el acceso a los recursos a los que podía acceder el 8086.

En 1985 nace el **80386**, que utiliza 32 bits (incluso en el bus de direcciones) con lo que se podían direccionar hasta 4 Gb de memoria. Sigue incluyendo el modo real para mantener la compatibilidad con versiones anteriores, pero esta vez permite el acceso a todos los recursos de la máquina.

Se puede conmutar entre ambos modos sin resetear. El modo protegido, además de la **memoria virtual**, permite utilizar **paginación** (con tamaño de página de 4 Kb). Se incluye el **modo 8086 virtual** que permitía usar, en modo protegido, programas del 8086. También se incorporaban **niveles de protección** (para diferenciar entre instrucciones del S.O. y de la aplicación).

Para mejorar el rendimiento, introduce el concepto de **segmentación** (referido a la ejecución de instrucciones). Se utilizan 6 etapas: acceso a memoria, búsqueda de instrucciones, decodificación, ejecución, proceso por la unidad de segmento y proceso por la unidad de paginación.

En el 1989 se lanza el **80486** que incluye el **coprocesador matemático** de serie (permitiendo operar con números reales de forma eficiente), pues en las versiones anteriores había que comprarlo aparte (añadiendo un siete a la nomenclatura: 80287, 80387, etc).

Para ganar en rendimiento, se dividen dos de las etapas de segmentación (decodificación y ejecución) en 10 etapas.

Se introduce la **memoria caché**, para disminuir el tiempo de espera del procesador. Dicha caché tiene 8 Kb para datos. El reloj del procesador llega a alcanzar los 100 Mhz.

En 1993 nace el **Pentium**, que es un computador superescalar, pues usa **dos cauces de ejecución** mediante duplicación. Se incorpora una caché de 8 Kb para instrucciones. Introduce un nuevo tipo de instrucciones llamadas **MMX** para mejorar el trabajo sobre múltiples datos en paralelo (SIMD: simple instruction multiple data).

Esto permite, por ejemplo, obtener un rápido procesamiento de imágenes (en vez de tratar píxel a píxel, se tratan grupos de píxeles). Se introduce un manejo de interrupciones avanzadas.

En 1995, Intel anuncia el concepto de **familia P6**, que utilizan procesadores con funcionalidad ampliada (Pentium Pro, Pentium II, Pentium II Xeon, Celeron, Pentium III y Pentium III Xeon).

El bus de direcciones pasa a 36 bits (permitiendo el uso de 64 Gb de Ram). Se usan **tres cauces de ejecución** (cada uno con 12 etapas). Se incorpora la **ejecución dinámica**, que consiste en analizar la forma en la que el código accede a los datos (análisis de microdatos) para realizar la ejecución de forma no secuencial de determinadas instrucciones (sin que ello afecte al programa). También se realizan la **predicción de saltos** y la **ejecución especulativa**.

El **Pentium II** añade un **segundo nivel de memoria caché** (de 256 Kb) y duplica el tamaño de las de primer nivel (16 Kb para código y 16 Kb para datos).

El calificativo **Xeon** está pensado para computadores multiprocesador y servidores de empresa.

Se crean las **instrucciones SSE**, que son instrucciones MMX aplicadas a flotantes.

El calificativo **Celeron** implica una reducción de la caché a 128 Kb para la caché de segundo nivel.

En el 2000 se lanza el **Pentium 4**, que utiliza la arquitectura **Netburst**. Se usan 20 etapas de ejecución. Las ALUs son el doble de rápidas que el procesador. Se mejora mucho la predicción de saltos y la ejecución especulativa (hasta 256 instrucciones por delante). Se incluye la **caché de micro operaciones**, situada a la salida de la decodificación y que almacena instrucciones ya ejecutadas (y decodificadas).

Se introduce el concepto de **Hyperthreading** que permite la ejecución de varias tareas simultáneamente de forma más eficiente. Esto se consigue ejecutando una sola instrucción para todas las tareas, cuando ésta se repite.

Soporte de la arquitectura IA-32 a los S.S.O.O. multitarea

Modos de funcionamiento

- **Protegido** = soporte multitarea y 8086 virtual.
- **Real** = 8086 ultra rápido.
- **Gestión de sistema** = permite llevar a cabo operaciones de gestión (energía, por ejemplo). Se entra en este modo cuando hay una instrucción especial.

Registros básicos del sistema

Registros de propósito general (32 bits): EAX, EBX, ECX (especializado en bucles), ESI, EDI, EBD, ESP (puntero de pila).

Por compatibilidad con el 8086 se puede usar la parte baja de los registros anteriores (convirtiéndose en AX, BX, CX, ...).

La parte baja de los cuatro primeros registros se puede dividir en dos trozos de 8 bits (AH, AL, BH, BL, ...).

Registros de segmento (16 bits): CS, DS, ES, FS, GS, SS.

Actúan como selectores para determinar la dirección de memoria donde se ubica un segmento de memoria.

Bits 0 y 1: **RPL** (requested privilege level); indica el nivel de privilegio requerido para acceder a este segmento.

Bit 2: **TI** (table indicator); indica la tabla que se usa para buscar con el selector. Si vale 0 se usa la **GDT** (global descriptor table) y si vale 1 se usa la **LDT** (local descriptor table).

Nota: Linux solo usa la GDT.

Los 13 bits restantes son el índice para buscar una entrada en la tabla. La entrada se llama descriptor. Esto nos permite tener $2^{13}=8192$ índices posibles.

Registro de estado y control EFLAGS (32 bits):

- Flag de zero : bit 6
- Flag de acarreo: bit 0
- Flag de signo: bit 7
- Flag de overflow: bit 11
- Flag de interrupción (IF): bit 9
- Flag de trap (TF): bit 8 (si es 1, se habilita el modo traza)
- Flags IOPL: bits 12 y 13 (nivel de privilegio de las instrucciones de e/s)

Nota: el trap refleja una toma de control del sistema operativo.

Registro EIP (32 bits): es el puntero de instrucción, su misión es apuntar a la dirección de memoria de la próxima instrucción a ejecutar.

Modelo de memoria

En esta arquitectura existen tres modelos de memoria:

- **Espacio de direcciones lógicas:** representa la forma en que los programas utilizan la memoria. Las direcciones están formadas por el registro de segmento + desplazamiento (16 bits + 32 bits = 48 bits).
- **Espacio de direcciones lineal:** representa la dirección lógica como una dirección de 32 bits.
- **Espacio de direcciones físicas:** representa la dirección de 32 bits de la memoria. De esta forma se traduce la dirección lineal a la real, teniendo en cuenta la RAM disponible.

Mecanismo de segmentación: calcula la dirección lineal del operando. Siempre se encuentra activo.

Mecanismo de paginación: traduce la dirección lineal a la dirección física. Se puede desactivar. Si se desactiva, la dirección lineal coincide con la física.

Mecanismo de segmentación (en profundidad)

Los descriptores de segmento tienen 64 bits y están divididos en tres trozos. Los 32 bits más bajos guardan la dirección base del segmento.

Del bit 32 al 51, se indica el tamaño del segmento. Con paginación hay 2^{20} (1 Mb) páginas máximo, siendo cada página de 4 Kb.

Del bit 52 al 64, se indican los atributos del segmento. Controlan el nivel de acceso al segmento, o el **DPL** (descriptor privilege level).

Nota: GDTR o LDTR es un registro que almacena la dirección de la tabla de descriptores.

Registros de sombra

Cada vez que accedemos a memoria, usamos dos accesos, uno para obtener el descriptor y otro a la dirección física. Para evitar esto, almacenamos el valor del descriptor de segmento en el registro de sombra, para que valores que estén en el mismo registro no tengan que buscar la dirección a memoria.

El tamaño del registro de sombra es de 64 bits. Hay tantos registros de sombra como de segmento (6).

Accesos a memoria

- Búsqueda de instrucciones (registro CS).

- Acceso a datos estáticos (registros DS, ES, FS, GS, siendo por defecto el DS).
- Acceso a datos dinámicos (registro SS).

ejemplo:

```
MOV ECX, [EDX]
MOV EAX, GS:[EBX+5]
POP EAX
```

La segmentación no se puede desactivar, pero sí puede reducirse su complejidad.

Modelo de memoria plano

Limita la segmentación a su mínima expresión (método usado en Windows y Linux). Solo utiliza la **GDT** para almacenar descriptores. Reduce el número de descriptores manteniendo el mínimo en cuatro.

Los descriptores toman como dirección base la 00000000h y tamaño máximo 4 Gb. Con esto conseguimos que el desplazamiento se convierta directamente en la dirección lineal.

Renunciar a la segmentación es renunciar a la protección de las direcciones de memoria.

Niveles de privilegio

Hay cuatro niveles de privilegio:

Nivel 0: núcleo del S.O.

Niveles 1 y 2: servicios del S.O.

Nivel 3: aplicaciones del usuario.

Nota: Windows y Linux solo usan el 0 y el 3.

El **RPL** fija los privilegios del segmento. Si el registro de segmento que se usa es el **CS**, el RPL pasa a denominarse **CPL** (current privilege level) y marca el privilegio del programa en ejecución.

El **DPL** (descriptor privilege level) establece quién puede acceder al segmento referenciado por el descriptor.

Los bits **IOPL** del registro **EFLAGS** marcan el privilegio necesario para acceder al espacio de direcciones de E/S. Establece quién puede ejecutar las instrucciones IN y OUT.

Nota: Windows y Linux lo ponen a 0 para que solo el operativo sea el que realice la entrada/salida.

Niveles de privilegio

1. Uso para evitar la ejecución de determinadas instrucciones (por ejemplo CLI).
2. Uso para controlar el acceso a las instrucciones de E/S (IOPL).
3. Cambio de segmento en ejecución (cambio de tarea).
 - Se está ejecutando basado en un registro. Segmento **CS** con un nivel de privilegio **CPL**.
 - Para cambiar de tarea, debe cambiarse **CS**, es decir, el nuevo **CS** tiene que tener un privilegio adecuado (igual o mayor).
 - Este cambio implica hacer comprobaciones. El **CPL** del registro segmento en ejecución, el **RPL** del valor que queremos cargar en el registro segmento **CS**. El **DPL** del descriptor asociado al valor del nuevo registro segmento.

4. Acceso a un nuevo selector del registro de segmento de datos (DS, ES, FS, GS, SS).
 - **CPL** del código en ejecución.
 - **RPL** del nuevo valor que se va a cargar en el registro de segmento.
 - **DPL**, nivel de privilegio del descriptor asociado al nuevo valor del registro de segmento.

Mecanismos de transferencia de control al S.O.

Hay que transferir el control al S.O. cuando:

- Se realiza una llamada al S.O.
- Se produce una interrupción.
- Se produce una excepción.

En cualquiera de los casos anteriores se llama a un **manejador** (programa especial) que forma parte del S.O.

Hay que guardar una tabla con las direcciones de los manejadores y asegurarnos de que tenemos suficiente privilegio para acceder a ella. Se llama **IDT** (tabla de descriptores de interrupción).

Se accede a la tabla a partir del registro **IDTR** (dirección base + tamaño). Cada entrada en la tabla anterior se llaman **puertas** (gates).

Tipos de puertas

- **Task gates**: usadas por Windows.
- **Interrupt gates**: se utilizan para acceder a manejadores de interrupciones y llamadas al S.O.
- **Trap gates**: se utilizan para acceder a manejadores de las excepciones.

Cada puerta consta de los siguientes campos:

Atributos (los 2 bytes altos): indica el tipo de puerta y el privilegio de acceso.

Selector (2 bytes): selector de segmento que contiene la dirección lógica donde comienza el manejador.

Desplazamiento (4 bits más bajos): desplazamiento de la dirección lógica donde está el manejador.

Tamaño de la **IDT**: $256 \times 8 \text{ bytes} = 256$ entradas como máximo.

Excepciones, interrupciones y llamadas al S.O. son exactamente iguales, solo varía la forma de obtener el número de puerta en la **IDT**.

Llamadas al S.O.

El número de la **IDT** forma parte de la instrucción.

ejemplo:

`INTm` donde `m` es el número de la IDT.

ejemplo en Windows:

`INT2Eh` proporciona la mayoría de los servicios al S.O.

Interrupciones

Son usadas por los periféricos o controladores de dispositivos. Los periféricos están conectados al bus de control y activan la línea **LINT** del controlador de interrupciones local (dentro del procesador) y suministra un dato que es el número de la IDT (también llamado número de vector de interrupción).

Si no existe **LINT** (486 y anteriores) se usa el **INTR** (enmascarable, colocando el bit IF a 0) y el **NMI** (no enmascarable, control del funcionamiento del sistema).

Excepciones

El número de la IDT la genera la CPU dependiendo de la situación anómala que lo produjo. Existen tres tipos:

- **Fallo** (fault): no se puede recuperar el estado.
- **Aborto** (abort): se corrompen las tablas del sistema y se produce un fallo general.
- **Trampa** (trap): el control vuelve a la siguiente instrucción ejecutada antes de saltar la excepción. Se suele usar para depurar.

Secuencia de pasos durante la transferencia de control

Consideramos una llamada a un servicio del S.O.

Dentro del código del programa: WT 2Eh

El programa se está ejecutando con privilegio de usuario (CPL =3).

1. El procesador obtiene el número de interrupción (número de puerta en la IDT).

excepciones – procesador
interrupción – controlador de interrupción
servicios – en la propia instrucción (2Eh)

2. Calcular la dirección de memoria donde está la puerta para consultar el número de interrupción (número de puerta) x 8.

ejemplo: $2Eh \times 8 = 170h$

3. Se leen los 8 bytes de la puerta IDT.

Se comprueba que la tarea tiene suficiente privilegio para poder llamar al servicio (comprueba los atributos de la IDT).

4. Se lee el selector de la IDT. Con este selector han de comprobarse los niveles de privilegio, para poder pasar a un nuevo selector. Se comprueba el CPL actual, el RPL del nuevo selector y el DPL del descriptor que apunta al nuevo selector. Si las comprobaciones son exitosas, se puede proseguir.
5. Se conmuta la pila (se pasa a utilizar la pila del operativo). Los registros SS y ESP de la tarea se guardan temporalmente y se sobrescriben con los valores guardados de SS y ESP de la pila del operativo. Los valores guardados van a la pila del operativo.
6. Se coloca el IF y el TF a cero (para que el operativo no se interrumpa).
7. Se transfiere el control al manejador:

- Se almacena en la pila el registro de estado EFLAGS y la dirección de retorno de CS y EIP.
- Se carga en CS y EIP la dirección donde comienza la primera instrucción del manejador.

- El EIP se consigue en la parte de desplazamiento de la IDT.
 - El CS se consigue en la parte de selector de la IDT.
8. Se ejecuta el código del manejador.
 9. Guardamos el estado de la tarea. Se almacena en una estructura interna del operativo los registros y se desapilan EIP, CS, EFLAGS, ESP y SS.
 10. El manejador realiza su trabajo.
 11. Se ejecuta el planificador que determina la tarea que ha de tomar el control cuando finalice el operativo.
 12. Se prepara el estado de la tarea para su recuperación. Se leen desde las estructuras de datos del operativo el estado de la tarea a recuperar (se leen los registros, se prepara la pila, se apilan SS, ESP, EFLAGS, CS y EIP).
 13. Se ejecuta la instrucción IRET (última del manejador). Se saca de la pila y se sobrescriben EIP, CS y EFLAGS. Se devuelve el control a la tarea.
 14. Se conmuta la pila. Se pasa a la pila de la tarea, desafilando ESP y SS.
 15. Continúa la ejecución normal de la tarea.

La jerarquía de memoria

Introducción

El propósito de la memoria es servir de almacén de instrucciones y datos. Para valorar el impacto del tiempo de acceso a memoria podemos utilizar esta fórmula:

$$\text{Frecuencia efectiva} = 1 / (\text{Periodo reloj procesador} + \text{Tiempo de acceso} * \text{razón de accesos a memoria})$$

Donde la razón de accesos a memoria es el número de operaciones de memoria sobre las totales. Como ejemplo tenemos que un procesador a 1 GHz con una memoria que tarde 60ms en realizar una operación de acceso y que 1 de cada 4 operaciones son de acceso a memoria, tenemos que la frecuencia real que obtenemos en el procesador ronda los 60 MHz.

Tecnologías de memoria

SRAM (static random access memory): Las celdas usan biestables. Es muy rápida (casi como la CPU).

DRAM (dynamic random access memory): Las celdas básicas usan condensadores. Es más lenta (hasta 100 veces más) y requiere un refresco periódico para no perder la información.

Almacenamiento magnético (discos y cintas): alta capacidad pero muy lentos (1.000.000 de veces más) que la SRAM).

El coste de la memoria viene dado por el tamaño de la celda y por la cantidad de energía necesaria para almacenar 1 bit.

Ejemplo:

Disco duro 'Seagate' ST3120026 A-R6 de 120 Gb (130€): 0,00108 €/Mb
 Tiempo de acceso = 10×10^6 nseg

Módulo de DRAM 'Kingston' KVR400x64 c25/512 (108€): 0,21 €/Mb
 Tiempor de acceso = 10 nseg

Comparativa de SRAM en dos P4 3,2Ghz: 100 €/Mb
 Tiempo de acceso = 1 nseg

Con los datos anteriores, construir 2 Gb de memoria costaría:

SRAM: $2048 \times 100 \text{ €/Mb} = 204.800 \text{ €}$

DRAM: $2048 \times 0,21 \text{ €/Mb} = 430 \text{ €}$

D.D: $2048 \times 0,00108 \text{ €/Mb} = 2 \text{ €}$

Concepto de jerarquía de memoria

Se busca una memoria de alta capacidad, alta velocidad y bajo coste. Esto se consigue combinando diferentes tecnologías (SRAM, DRAM, Discos). La jerarquía funciona si conocemos la probabilidad de acceso futura a los datos. Estos datos más frecuentes se colocan en la SRAM. Los datos que no quepan ahí, los guardamos en la DRAM y el resto en el disco.

Para averiguar la frecuencia de acceso se usa el principio de localidad. Este principio se da en los programas ejecutados en un computador y puede ser:

- **Localidad temporal:** si accedemos a una dirección de memoria, es probable que en un corto periodo de tiempo volvamos a acceder a ella.
- **Localidad espacial:** si accedemos a una dirección de memoria, es probable que accedamos a posiciones muy próximas.

Éxito (hit): cuando se busca un dato en un nivel de memoria y se encuentra.

Fallo (miss): cuando se busca un dato en un nivel de memoria y no se encuentra.

Si no se encuentra un dato en un nivel, se pide al nivel superior.

Ejercicio: se construye un sistema de 2 Gb: 1 Mb de SRAM, 512 de DRAM y 1,5 Gb de disco.

¿Coste? ¿Tiempo de acceso?

Utilizamos los datos del ejercicio anterior.

Coste = $1 \times 100 + 512 \times 0,21 + 1500 \times 0,00108 = 209 \text{ €}$

Probabilidad de acceso: SRAM = 99,9%, DRAM = 99,9%

Tamaño del bloque: 64 bytes para memoria principal, 4k para disco.

$T_{\text{acceso}} = T_{\text{cache}} \times \text{Aciertos}_{\text{cache}} + (1 - \text{Aciertos}_{\text{cache}}) T_{\text{mp_disco}} \times \text{Bloque}_{\text{mp}}$
 $= 1 \times 0,999 + (1 - 0,999) \times 11 \times 64 = 1,703 \text{ nseg.}$

$T_{\text{mp_disco}} = T_{\text{mp}} \times \text{Aciertos}_{\text{mp}} + (1 - \text{Aciertos}_{\text{mp}}) T_{\text{disco}} \times \text{Bloque}_{\text{disco}}$
 $= 10 \times 0,999 + (1 - 0,999) \times 10^7 = 11 \text{ nseg.}$

Memoria caché

Conceptos preliminares

Es el **primer nivel** de memoria del computador. Utiliza tecnología **SRAM** que es rápida pero pequeña. La CPU siempre pide el dato a la memoria caché. Si ésta lo tiene, lo sirve. Si no lo tiene, lo pide al nivel superior de la jerarquía y cuando lo recibe lo envía a la CPU. Mientras sucede esto, la CPU esta parada.

La memoria caché se organiza en **bloques**, que se definen como un conjunto de posiciones consecutivas en memoria (palabras de memoria).

Nota: nosotros vamos a considerar que una palabra de memoria es 1 byte.

Desde el punto de vista de la caché, la memoria principal es un almacén de bloques.

Ejemplo:

Memoria cache de 32 bytes, tamaño bloque = 4 bytes, es decir 8 bloques.

Memoria principal de 256 bytes, es decir 64 bloques.

La organización en bloques define dos campos en la dirección:

- Número de bloque (6 bits).
- Desplazamiento dentro del bloque (los 2 bits menos significativos).

Aspectos importantes

Estrategia de correspondencia: determina cómo se ocupan los bloques de la caché. Para ello necesitamos controlar la identificación (el bloque de memoria que se corresponde con el de caché) y la ubicación (la posición que ocupa ese bloque de memoria en la caché).

Estrategia de reemplazo: determina la posición de un bloque cuando todas las posiciones están ocupadas.

Estrategia de escritura: gestiona la validez de la información cuando ésta es modificada.

Estrategias de correspondencia

Correspondencia directa: es la más simple. Cada bloque de caché se calcula con la siguiente operación

Bloque caché = (bloque memoria principal)%(nº bloques de caché)

Normalmente el número de bloques es 2^x . Si es así, el bloque viene fijado por los x bits menos significativos del número de bloque. Los tres bits más significativos son un identificador de bloque. Necesitamos un bit adicional para comprobar si la información almacenada es válida o no.

Ejemplo:

Lectura del dato de la dirección 01001110b

Suponemos que el bit de validez está a 0 en el bloque 3 y necesitamos traer la instrucción de memoria.

Copiamos el bloque de memoria principal a la caché.

Nombramos el bloque con los tres bits más significativos (010) y situamos el bit de validez a 1.

Repetimos el intento de búsqueda en la caché y obtenemos un acierto.

Si intentamos leer la 01001111 la habíamos subido en el paso anterior, luego tenemos un acierto. Si intentamos la 01010000 obtenemos un fallo y hemos de repetir la operación.

Ejemplo:

Bucle:

Acceso a 01011100

...

Acceso a 10111110

Repetir 10.000 veces.

Tienen igual posición de etiqueta pero distinto contenido, se accede a memoria 9.999 veces.

Correspondencia (totalmente) asociativa: los dos bits menos significativos de la dirección marcan el desplazamiento. Los seis restantes marcan la etiqueta. No hay número de bloque, tenemos que ir comparando uno a uno cada bloque. Esto otorga mayor flexibilidad (no se desperdician bloques) pero es mucho más lenta.

Correspondencia asociativa por conjuntos: definimos el concepto de conjunto como grupo de bloques. Para elegir el conjunto usamos correspondencia directa. Se usa correspondencia totalmente asociativa dentro del conjunto.

Ejemplo:

01001100 acierto

11001110 fallo

Estrategias de reemplazo

Deciden el bloque de caché a intercambiar cuando todos los bloques posibles están ocupados.

Nota: en la correspondencia directa no existe el problema de reemplazo.

LRU (last recent used): reemplaza el bloque que hace más tiempo que no se usa.

Reemplazo aleatorio: se elige un bloque al azar. Es muy simple pero no da unos resultados excesivamente mediocres.

Estrategias de escritura

Escritura directa o a través (write-through): cuando se modifica la caché, se modifica la memoria principal.

Escritura diferida o post-escritura (write-back): no se modifica la memoria principal cuando se modifica la caché. Cuando el bloque de caché es reemplazado, escribimos en memoria principal.

La última ofrece un mejor rendimiento pero necesita hardware adicional. Se utiliza un bit en cada bloque, llamado bit de **dirty**. Si está a 1, entonces la información del bloque se escribirá en memoria principal cuando sea reemplazado.

El problema de la coherencia

La solución que aportan las estrategias anteriores solo sirve cuando el procesador es el único elemento que accede a memoria principal, pero puede ser un periférico el que acceda.

Si el periférico está mapeado en memoria, podemos marcar esa posición como **no cacheable**.

Si el periférico tiene capacidad **DMA** (acceso directo a memoria), un bloque cacheado de memoria principal puede ser modificado por el periférico causando incoherencia pues la CPU trabaja con información obsoleta.

El procesador podría modificar en la caché un bloque de memoria utilizando la estrategia de escritura diferida. Un periférico leería dicho bloque de memoria principal recibiendo información obsoleta.

No podemos marcar toda la memoria principal como no cacheable, necesitamos otra solución. Para solucionar esto se usa el **espionaje del bus o snooping**. El controlador de caché observa continuamente las líneas de lectura y escritura de memoria principal y las líneas del bus de direcciones.

Ejemplo:

Un computador posee un sistema jerárquico de memoria formado por un sistema principal de 1024 bytes y una memoria caché de 64 bytes. La caché funciona con una estrategia de correspondencia directa, estrategia de escritura diferida y está organizada en 8 bloques.

En un momento dado, las etiquetas, bits de validez y bits de dirty reflejan estos valores:

Nº de bloque	Bit de Validez	Bit de Dirty	Etiqueta
0	1	1	0010
1	X	X	XXXX
2	1	0	1101
3	1	0	0110
4	X	X	XXXX
5	1	0	0100
6	1	0	0000
7	1	1	0111

Contesta las siguientes preguntas:

a) Si el procesador realiza tres operaciones de lectura consecutivas sobre 350h, 0EBh y 0E8h, en este orden, ¿Qué bloque de memoria principal resultarán reemplazados?

350h = 0000001101010000b

0EBh = 000000001011011b

0E8h = 000000001011000b

Tenemos 1024 posiciones. Para direccionarlas necesitamos 2^{10} , es decir 10 bits.

350h = 1101010000

0EBh = 0011101011

0E8h = 0011101000

Tamaño de bloque = 64 bytes / 8 bloques = 8 bytes/bloque.

8 bytes = 2^3 , es decir, 3 bits para el desplazamiento.

8 bloques = 2^3 , es decir, 3 bits para el bloque.

1101 010 000 acierto

0011 101 011 fallo
0011 101 000 acierto

La dirección de memoria principal que vamos a reemplazar es 0100101 = 25h = 37d
Reemplazamos la etiqueta 0100 por la nueva: 0011. Solo se modifica un bloque, el 37.

b) Partiendo del estado inicial de la caché, el sistema de memoria recibe tres peticiones de lectura sobre las direcciones 259h, 034h, 1F9h. Dichas peticiones son realizadas por un periférico con capacidad DMA. Se desea saber qué bloque(s) de memoria principal resultarán actualizados como consecuencia de dichas operaciones de lectura.

1001 011 011 no cacheado, el periférico puede leer sin problemas.
0000 110 100 está en caché, pero su bit de dirty está a 0.
0111 111 001 hay que actualizar este bloque por 0111111 = 3Fh = 63d

Organización de la memoria caché

Según el número de niveles de caché:

Los niveles de caché se denominan L1, L2, etc. En cada nivel superior se gana en capacidad pero se pierde en tiempo de acceso.

Según la separación entre caches de datos y código:

- **Caché unificada** (única para datos y código): se aprovecha mejor el espacio y de forma más eficiente.
- **Caché dividida, separada ó partida**: separa la memoria caché en dos partes, una para código y otra para datos. Se aprovecha poco espacio y pueden existir bloques vacíos.

Memoria caché en la arquitectura IA-32/PC

Todos los procesadores modernos disponen de dos niveles de caché. El primero (L1) es una caché dividida, y el segundo (L2) es una caché unificada. Ambos niveles usan una estrategia asociativa por conjuntos y se encuentran en el interior del procesador para mejorar la velocidad de transferencia.

En particular, el procesador **AMD Athlon**:

La caché L1 de datos tiene un tamaño de 64 Kb, su tamaño de línea (bloque) es de 64 bytes. Su número de vías (número de bloques por conjunto) es de 4. Tiene 256 conjuntos.

La caché L1 de código es idéntica a la anterior.

La caché L2 unificada tiene un tamaño de 256 kb, un tamaño de línea de 64 bytes (para facilitar el intercambio de datos entre coches), 16 vías y 256 conjuntos.

La memoria principal

La comunicación entre memoria principal y la memoria caché se organiza con bloques de tamaño x bytes. La transferencia de información se realiza a través del bus de datos.

Etapas en el proceso de lectura en memoria principal

Número de accesos = tamaño bloque / ancho de bus de datos.

El proceso de transferencia (acceso de lectura a memoria principal) tiene tres etapas:

1. Envío a la memoria principal de la dirección del bloque donde está el dato.
2. Búsqueda del dato en esa dirección de la memoria principal.
3. Transmisión del dato desde la memoria principal a la memoria caché a través del bus de datos.

Ejemplo: coste de fallo de caché.

Tamaño de bloque: 32 bytes.

Ancho del bus de datos: 8 bytes.

Coste temporal de operaciones: (1 ciclo + 3 ciclos + 1 ciclo).

Coste para transferir: $(1+3+1) \times (32/8) = 20$ ciclos por acceso.

Coste temporal de la actualización de un bloque de memoria principal

La memoria principal se actualiza cada vez que se modifica la caché (write-through) o cada vez que se reemplaza un bloque modificado (write-back). También se actualiza cuando un periférico accede a un bloque cacheado y modificado en la caché con estrategia diferida.

Etapas en el proceso de escritura en memoria principal

1. Enviamos la dirección en que se desea escribir a través del bus de direcciones.
2. Transmitimos, a través del bus de datos la información a almacenar.
3. Escritura de la información enviada, en la memoria.

Ejemplo:

Tamaño del bloque: 32 bytes.

Ancho del bus: 8 bytes.

Coste temporal de las operaciones: (1 ciclo + 2 ciclos + 4 ciclos).

Coste total de actualización: $(1+1+4) \times (32/8) = 24$ ciclos.

Mejoras en la organización de la memoria principal

Formas de actuación para mejorar el acceso a memoria principal:

- **Mejoras tecnológicas** en la memoria principal: celdas de memoria cada vez más rápidas.
- **Mejoras en la organización** de la memoria principal.

Para optimizar los accesos podemos realizar las siguientes mejoras:

- **Localidad**: se intercambia información entre memoria caché y memoria principal en bloques.
- Los dispositivos físicos de memoria principal están organizados en **grupos** (bancos).

Memoria no entrelazada: no se permiten accesos simultáneos a distintos bancos.

Memoria entrelazada: se permiten intercalar los accesos a distintos bancos.

Ejemplo:

Tamaño del bloque: 32 bits.
Tamaño del bus: 8 bytes.

Suponemos lectura sobre F0000000 a F000001Fh.
Suponemos que tenemos dos dispositivos de memoria (bancos): A y B.
A la hora de devolver un dato, los bloques han de ir de uno en uno, pues el tamaño del bus de datos es finito.

Para solucionar este problema, podemos ampliar el bus de datos. De esta forma, varios dispositivos pueden enviar datos simultáneamente. Pero este cambio es muy costoso, pues requiere cambios en el procesador y además, tantas líneas producirían interferencia unas con otras.

Acceso mediante ráfagas

El controlador de caché solo envía la primera dirección de memoria y la memoria principal se encarga de generar las siguientes. De esta forma los datos son servidos de forma mucho más eficiente.

La memoria virtual

Introducción

La memoria virtual es la técnica que permite que la memoria principal actúe como memoria caché de los dispositivos de almacenamiento secundario.

Funcionamiento

La CPU trabaja con direcciones lógicas (espacio de direcciones lógico o virtual). La memoria trabaja con direcciones físicas (espacio de direcciones físico o real). La traducción de una a otra, teniendo en cuenta la memoria instalada en el sistema la realiza la **MMU** (memory management unit).

Ventajas

Cada tarea dispone de todo el espacio de direcciones virtuales en exclusiva, es decir, se puede escribir en cualquier posición del espacio virtual. La MMU es la encargada de escribir en cualquier posición del espacio virtual.

Asimismo, la **MMU** se encarga de asignar direcciones físicas adecuadas a las direcciones virtuales de cada tarea. En el caso particular de que existan dos MMU, una traducirá las direcciones lógicas a lineales y otra las lineales a físicas.

Funcionamiento de la MMU

La MMU utiliza una **tabla de traducción**. En ella, a cada dirección virtual se le asigna su correspondiente dirección física. Cada tarea tiene su propia tabla de traducciones.

Objetivos

- **Ampliación** de la capacidad de memoria.
 - Nos permite ejecutar programas más grandes que la memoria física instalada.
 - Nos permite ejecutar varios programas que juntos ocupan más que la memoria física instalada.

Esto se consigue usando una zona de disco como zona de memoria. En Windows se denomina **archivo de paginación**, en Linux, **área de swap**.

En Windows el archivo de paginación tiene el doble de tamaño que la memoria principal. Este archivo no es la única zona de memoria en disco, pues los programas ejecutables en disco actúan como archivos de paginación.

- **Protección** de memoria.
 - Mediante la adecuada programación en las tablas de traducción se impide que se solapen áreas de memoria física de dos tareas.
 - Incorporar en las entradas de la tabla de traducción un conjunto de bits que controlen los privilegios de acceso.
- **Compartición** de memoria.
 - Tareas que comparten información (ejemplo: el portapapeles de Windows).
 - Compartición de código ejecutable (ejemplo: DLLs).

La compartición de memoria la controla el sistema operativo programando adecuadamente las tablas de traducción de las tareas para que dos instrucciones lógicas hagan referencia a la misma dirección física.

- **Simplificación** de las herramientas de desarrollo y carga.

Cada tarea tiene su propia tabla de traducción que le permite disponer de todo el espacio de direcciones virtuales. En el proceso de carga, el sistema operativo programa las tablas de traducción de acuerdo a los huecos de memoria. Esto permite que un programa no ocupe un bloque continuo.

Estrategias de memoria virtual: paginación y segmentación

La tabla de direcciones, en caso de querer direccionar todo el espacio de direcciones ocuparía lo mismo que éste último. Tenemos que dividir la tabla en trozos más pequeños.

La asignación de direcciones se hace por bloques. A cada bloque de direcciones virtuales se le hace corresponder un bloque de direcciones físicas. Este mecanismo aumenta la eficacia de la tabla de traducción, reduce su tamaño y optimiza las búsquedas.

Para ello podemos usar distintos **tamaños de bloque**:

- **Tamaño fijo**: usa los bloques necesarios para cubrir los requerimientos del programa. Genera fragmentación interna.
- **Tamaño variable**: asigna la memoria justa que requiere cada proceso. Genera fragmentación externa.

Cuando un programa termina, un tamaño variable genera huecos libres que son difíciles de gestionar. Si utilizamos un tamaño fijo, el máximo espacio desperdiciado es de un bloque.

En la **memoria virtual paginada**, los bloques son de tamaño fijo y reciben el nombre de páginas. El tamaño de página se fija por hardware. Comúnmente se utilizan 4kb.

En cambio, en la **memoria virtual segmentada**, los bloques son de tamaño variable y reciben el nombre de segmentos.

La paginación en IA-32 nos ofrece un tamaño total de 4 Gb de direcciones de memoria y un bloque de 4kb. Este mecanismo es totalmente transparente al programador. Es mucho más sencilla de implantar. El acceso a memoria es más eficiente.

La dirección virtual se descompone en dos campos denominados página y desplazamiento.

La tabla de páginas

Es una estructura utilizada para convertir direcciones virtuales en direcciones físicas. Asigna páginas virtuales a marcos de página. El marco de página puede estar asociado a memoria principal o puede corresponder con una zona de disco.

Existe **una tabla de páginas por tarea**. Las tablas de páginas se guardan en memoria y son gestionadas por el S.O.

La posición en memoria de la tabla de páginas se obtiene de un registro de control de la CPU, llamado **registro de tabla de páginas**.

Cada entrada de la tabla de páginas tiene los siguientes campos:

- **Bit de presencia:** indica si dicha entrada esta cargada en memoria o no.
- **Marco de página o desplazamiento:** Si el bit de presencia es 1, contiene la dirección del marco de página que contiene la página a la que se desea acceder. Si es 0, indica el desplazamiento en el archivo de paginación. Si tiene otro valor, la página no tiene un almacenamiento asociado.
- **Bits de protección:** indican quién tiene acceso y para qué.
- **Bits de estado:** indican si una página ha sido modificada, etc.
- **Bits de acceso:** contabilizan los accesos a la página.

Ubicación en memoria de las tareas y el sistema operativo

Las tareas habitualmente hacen llamadas a servicios del sistema operativo. Parte del código del operativo formará parte de cualquier tarea. Para no tener páginas repetidas en memoria que pertenezcan al sistema operativo, se programa la tabla de páginas para que se **compartan** dichas páginas entre las tareas que la usan.

Protección de memoria

Existen tres mecanismos de protección de memoria:

1. **Programación adecuada de la tabla de páginas:** mantener la independencia de los espacios de direcciones.
2. **Según el tipo de acceso:** lectura, escritura, ejecución, etc.
3. **Según el nivel de privilegio:** usuario o supervisor.

Los dos últimos mecanismos utilizan bits específicos de la tabla de páginas.

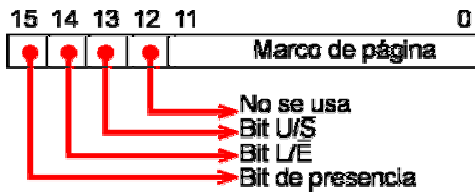
Nota: en la IA-32 los 4Gb de memoria se dividen en 64kb para la comprobación de punteros nulos. Los 2Gb siguientes se asignan a las tareas y los 2Gb finales son utilizados por el sistema operativo.

Compartición de memoria

La zona de memoria compartida se crea a petición de una tarea, mediante servicios específicos del operativo.

Ejercicio: un computador trabaja con direcciones virtuales de 32 bits, tamaño de páginas de 4k y direccionamiento al byte.

Cada entrada de la tabla de páginas tiene los siguientes campos:



En un momento dado, las tablas de páginas de dos tareas son las siguientes:

Tarea 1

	P	L/E	U/S	-	Marco
00100	1	1	1	-	014
00101	1	0	1	-	070
00102	1	0	1	-	071
...	-	...
07000	0	0	1	-	despl. X
07001	0	0	1	-	despl. Y
07002	0	0	0	-	inválido
...	-	...
10001	1	1	0	-	14F
10002	1	0	0	-	150
10003	0	1	0	-	despl. Z
...	-	...
B000F	0	0	0	-	inválido
B0010	1	0	0	-	Inválido

Tarea 2

	P	L/E	U/S	-	Marco
00100	1	1	1	-	025
...	-	...
00201	1	0	1	-	070
00202	1	0	1	-	085
...	-	...
05004	1	0	1	-	A2F
05005	0	0	1	-	despl. W
05006	0	0	0	-	inválido
...	-	...
A400	1	1	0	-	14F
A4001	1	0	0	-	150
A4002	0	1	0	-	despl. Z
...	-	...
C2000	0	0	0	-	Inválido
C2001	0	0	0	-	inválido

Responder a las siguientes preguntas:

1) ¿Tamaño máximo del espacio de direcciones físicas?

La dirección virtual tiene 32 bits. Los 12 bits más bajos se usan para el desplazamiento de la página por lo que los 20 bits restantes se usan para buscar el número de la página.

La dirección física tiene los 12 bits de desplazamiento igual que la dirección virtual. Además tiene otros 12 bits para el marco de la página. Por lo tanto $12 + 12 = 24$ bits.

El resultado es $2^{24} = 16$ Mbytes.

2) Si el sistema operativo establece que las páginas de código son de solo lectura y las de datos de lectura y escritura, entonces:

- ¿Número de páginas asignadas a cada tarea?
Tarea 1 = 9, Tarea 2 = 8.
- ¿Cuántas páginas en disco tiene cada tarea?
Tarea 1 = 3, Tarea 2 = 2.
- ¿Cuántas páginas están asociadas al sistema operativo en cada tarea?
Tarea 1 = 4, Tarea 2 = 3.
- ¿Cuántas páginas de datos comparten las tareas?
Comparten un página (070).
- ¿Cuántas páginas del sistema operativo comparten las tareas?
Comparten dos páginas (14F y 150).

Ejercicio:

Dado un sistema de memoria virtual con direcciones virtuales de 40 bits, un tamaño de página de 16 kb y direcciones físicas de 36 bits, contesta las siguientes preguntas:

a) ¿Tamaño total de la tabla de páginas para cada tarea, suponiendo que los bits de validez, protección, etc. ocupan 4 bits y que todas las páginas virtuales están en uso?

La tabla de páginas tiene dos dimensiones. Por un lado, tenemos los bits que ocupa cada entrada y por otro el número de entradas total.

Las direcciones virtuales tienen 14 bits de desplazamiento. Esos 14 bits se conservan en las direcciones físicas. Por lo tanto hay otros 22 bits para completar los 36 con lo cual el número de bits de cada entrada es de 26 bits (22 + 4 de los bits de estado).

El número de entradas es de 2^{26} , luego el tamaño total de la tabla será de $2^{26} \times 26 = 1664$ Mbits = 208,4 Mbytes.

El TLB (translation lookaside buffer o búfer de traducción)

Este búfer se encarga de **facilitar la traducción** de direcciones virtuales a direcciones lógicas. Evitamos hacer dos accesos a memoria (uno a la tabla de páginas y otro a la dirección en sí) para instrucciones pertenecientes a la misma página, almacenando en la TLB las entradas de la tabla usadas más recientemente.

El TLB es una memoria caché **totalmente asociativa**. La gestión de el TLB la realiza el operativo, pues tiene un coste parecido a hacerlo por hardware.

El sistema operativo **almacena nuevas entradas** en el TLB, **marca las páginas no útiles** como inválidas y **gestiona el reemplazo** de entrada. Todas estas acciones se realizan mediante instrucciones especiales.

El TLB y los cambios de contexto

El registro de la tabla de páginas almacena la dirección física del comienzo de la tabla de páginas de la tarea en ejecución. Si utilizamos multitarea, las tareas y por tanto sus tablas de páginas son cambiadas frecuentemente. Al cambio de una tarea a otra se le denomina **cambio de contexto**.

Esta acción implica el cambio del registro de la tabla de páginas a para apuntar a la tabla de páginas de la nueva tarea que va a ejecutarse. Debido a estos cambios de contexto, las entradas de la TLB han de actualizarse rápidamente, pues quedan inválidos al cambiar la tarea en ejecución.

Esto provoca ineficiencia debido a que las tareas pierden entradas que quizá en una próxima ejecución vuelvan a usar. Para solucionar este problema se le añaden bits adicionales a cada entrada del TLB que identifican la tarea a la que pertenece dicha entrada.

Con esta solución, las entradas no se invalidan y pueden estar disponibles para posteriores ejecuciones de la tarea (suponiendo que no resulten reemplazadas). La **técnica de identificación de tareas** permite definir algunas entradas como globales (por ejemplo, entradas al sistema operativo) comunes a todas las tareas. Estas entradas del TLB estarían disponibles casi siempre.

La solución anterior implica aumentar la capacidad del TLB para evitar los continuos reemplazos de entradas. De todas formas, cuando se alcance el límite de capacidad del búfer deberán reemplazarse entradas, generalmente con algoritmos de tipo **LRU**.

Existencia de la memoria caché

El contar con una memoria caché implica que los accesos a memoria principal se transforman en dos accesos a memoria caché. La memoria caché sirve para almacenar tanto entradas de la tabla de páginas como datos y código de instrucciones.

El TLB y la caché **comparten información** de forma similar a las caches divididas, aunque presenta un rendimiento menor que la caché unificada de tamaño equivalente, en cambio aumenta la rapidez al permitir accesos paralelos.

El TLB se referencia con direcciones virtuales, sin embargo los accesos a la caché se realizan con direcciones físicas. Esto genera un conflicto si queremos usar ambas para cachear las entradas de la tabla de páginas.

Una solución muy usada es que la TLB y la caché trabajen **segmentadas** en vez de en paralelo, es decir, la dirección entra al TLB y sale una dirección física que es enviada a la caché para obtener el dato.

Otra solución consiste en utilizar una caché que funcione con direcciones virtuales, pero es mucho más complejo, sobretodo al gestionar datos compartidos (que tendrían una dirección física y múltiples virtuales) porque podría estar cacheado varias veces y no actualizarse correctamente.

El manejador de fallo de página

El fallo de página se produce cuando la dirección virtual de una página no está almacenada en memoria física. Para detectar este caso se utiliza el bit de presencia que se encontraría desactivado. Esa instrucción genera una excepción denominada **excepción de fallo de página** que llama a una rutina del sistema operativo denominada **manejador de fallo de página**.

El manejador realiza una serie de acciones:

1. Determina la causa del fallo de página.
 - Dirección virtual sin almacenamiento asignado (inválidas).
 - Dirección virtual con almacenamiento asignado en disco.
2. Si la dirección pertenece a una página virtual sin almacenamiento asignado, el sistema operativo provoca una excepción por violación de acceso a memoria y se termina el programa.
3. Si la dirección tiene almacenamiento en disco, se llevan a cabo acciones para cargar la página del disco a memoria física.
 - Se busca espacio en memoria física para la página de disco que se desea cargar.
 - Si el working set está agotado (todas las páginas de memoria física ocupadas) se debe ejecutar un algoritmo de reemplazo para buscar un hueco (algoritmo tipo LRU).
 - Se comprueba si la página a reemplazar ha sido modificada pero no guardada (comprobando el bit de dirty). Si no lo ha sido, se debe guardar a disco antes de reemplazar. Si, por el contrario, ha sido guardada, se reemplaza sin más.
 - Se carga la página de disco al marco de página liberado, actualizando la entrada de la tabla de páginas correspondiente.
 - El manejador de fallo de página retorna a la instrucción, que ahora sí puede ejecutarse, y el programa continúa.

Paginación en la arquitectura IA-32

Existen tres espacios de direcciones:

1. El **espacio de direcciones lógico o virtual** (selector de segmento + desplazamiento) que es capaz de direccionar hasta 64 Tbytes en zonas de memoria estancas denominadas segmentos.
2. El **espacio de direcciones lineales** utiliza direcciones de 32 bits y permite direccionar 4 Gbytes.
3. El **espacio de direcciones físico o real** que utiliza direcciones de 32 bits y permite direccionar 4 Gbytes definidos por los buses del sistema.

Para desactivar la paginación se usa un bit de control. Con este sistema, la dirección lineal corresponderá con la dirección física.

Para evitar la complejidad de la segmentación se usa siempre el selector 0000. En este caso, el desplazamiento de la dirección virtual coincide con la dirección lineal. Esta arquitectura permite usar distintos tamaños de páginas pero, en la práctica, se suele usar un tamaño de 4 kb.

La tabla de páginas en la arquitectura IA-32

Teóricamente, cada tabla ocuparía:

$$4 \text{ bytes} \times 2^{20} \text{ entradas} = 4 \text{ Mbytes}$$

Esta forma de hacer la tabla de páginas es inviable, pues consume demasiada memoria. En la práctica se utilizan **tablas de páginas de doble traducción** (de dos niveles).

El campo de número de página virtual se divide en dos trozos:

- **Directorio de páginas** (10 bits) que indica la tabla de páginas de segundo nivel en la que está la página.
- **Posición** (10 bits) de la dirección en la tabla de páginas de segundo nivel.

El tamaño de esta estructura de dos niveles es:

$$4 \text{ kb} + 4 \text{ kb} \times 1024 = 4,004 \text{ Mbytes}$$

Una entrada en el directorio de páginas apunta a una tabla de páginas (4 kb) y cubre un rango de memoria de $2^{22} = 1024$ (entradas de la tabla) x 4 K (tamaño de página).

En la práctica se usan pocas entradas del directorio de páginas. Las tablas de páginas pueden ser, a su vez, paginadas y llevadas al archivo de paginación para dejar libre un área de memoria. Sin embargo, el directorio de páginas debe estar **siempre en memoria**. La dirección del comienzo de dicho directorio se almacena en el registro CR3 o **registro de tabla de páginas**.

Campos de una entrada en las tablas de páginas de la arquitectura IA-32

Hay 32 bits para cada entrada. Existen dos tipos de entradas: **EDP** (entrada de directorio de páginas) y **ETP** (entrada de tabla de páginas).

Para las EDP y ETP, los bits son **comunes**, pero la información de la EDP se aplica a la memoria ocupada por la tabla de páginas y la información de la ETP se aplica al marco de página asociado a la página virtual traducida.

Los distintos campos son los siguientes:

- **Marco de página**: 20 bits que codifican el marco de página asociado a la página virtual.
- **Bit P**: bit de presencia.
- **Bit W**: acceso de lectura o lectura y escritura.
- **Bit U**: permisos de acceso (usuario o supervisor).
- **Bit PWT**: indica el tipo de política de escritura en la caché (escritura directa o diferida).
- **Bit PCD**: permite deshabilitar la caché para la entrada actual.
- **Bit A**: bit de página accedida recientemente.
- **Bit D**: bit de página modificada (dirty).

También hay dos bits vacíos y cuatro exclusivos para el sistema operativo, que suelen ser usados para **identificar la tarea**.

Si el bit de presencia está a 0, la página no está cargada en memoria y los bits restantes pueden ser usados por el sistema operativo. Si está a 1, la entrada EDP indica dónde se almacena la tabla de páginas. Si la entrada es de la ETP, indica la dirección de almacenamiento del marco de página.

El acceso a una dirección virtual concreta puede provocar dos tipos de fallos de página. Si la entrada de la EDP tiene su bit de presencia a 0, hay que buscar a disco la tabla de páginas. Si la entrada de la ETP tiene su bit de presencia a 0, hay que buscar en disco la página asociada. Si la página no tiene almacenamiento asociado, ocurre una **violación de acceso** a memoria.

Tablas de páginas locales y globales

Las tablas de páginas locales son aquellas que pertenecen a una tarea en específica. Las tablas de páginas globales son aquellas asociadas al sistema operativo y son válidas para todas las tareas.

Utilizando ambas, se mejora el rendimiento, pues evitamos:

- La repetición de entradas en el TLB.
- Múltiples cambios en las tablas de páginas de las tareas al modificar páginas del sistema operativo.

Ejercicio:

Se tiene una arquitectura de 16 bits con memoria virtual paginada, con páginas de 1k palabras. Tanto las direcciones virtuales como las físicas son de 16 bits. Para almacenar la tabla de páginas se usa la misma técnica que Intel, es decir, doble nivel de traslación mediante directorio de páginas y tabla de páginas, siendo la interpretación de una dirección virtual como se indica en la siguiente figura:

Ya que tanto el directorio de páginas como la tabla de páginas se almacenan en memoria física, sus entradas son de 16 bits, si bien no se usan todos. El significado de cada bit se muestra en la figura siguiente:

Tanto el directorio de páginas como la tabla de páginas tienen igual tipo de entradas aunque en el directorio de páginas el marco se refiere a dónde encontrar la tabla de páginas, mientras que en la tabla de páginas se refiere a cómo obtener la dirección física final.

En esta arquitectura, el directorio de páginas se almacena en la dirección 0. Los contenidos de la memoria física en un instante dado se muestran en la siguiente figura, en la que las posiciones que no se muestran contienen ceros:

a) Si la CPU emite la dirección virtual 08AB, ¿Qué dirección física emitirá la MMU? (puedes responder fallo de página si crees que esto es lo que ocurre)

08AB = [000][0 10][00 1010 1011]

La primera posición representa el índice en el directorio de páginas, luego en la dirección 000 hay el dato 0401.

0401 = [0000 01][00 0000 00][0][1]

La tabla de páginas correspondiente está cargada en memoria (bit de presencia a 1).

La primera dirección de la tabla de páginas es la

0000 0100 0000 0000 = 0400

El índice TP nos dice que nuestra entrada se encuentra en la posición 2 de la tabla de páginas por lo que:

0400 + 2 = 0402

En la dirección 0402 hay el dato 2401 que en binario es:

```
[0010 01][00 1010 10][1][1]
```

Podemos ver que el marco se encuentra en memoria principal. Utilizamos la dirección del marco y le añadimos el desplazamiento:

```
[0010 01][00 1010 1011] = 24AB
```

La dirección que emite la MMU es la 24AB!

b) ¿Cuántas posiciones de memoria física se usa en total, en este caso, para almacenar todas la estructuras de datos relativas a la paginación? (ten en cuenta que es posible que no todas las tablas de páginas estén siendo usadas)

El sistema de entrada/salida

Introducción

Se denomina sistema de entrada/salida al **conjunto de interfaces de todos los periféricos del sistema**. Lo normal es que cada periférico esté asociado a un único interfaz, aunque puede darse el caso de que un interfaz atienda a más de un periférico.

Ubicación del sistema de entrada/salida en los espacios de direcciones

El sistema de e/s **forma parte del espacio de direcciones** de la CPU, es decir, todo el conjunto de direcciones que la CPU puede formar.

Hay dos formas de realizar esto:

1. Situar el sistema de e/s en el espacio de direcciones denominado espacio de direcciones de memoria. La CPU, al utilizar una posición de memoria no acude a la memoria principal si no a los registros del periférico. Este sistema es usado, entre otros, por grandes servidores.
2. En la arquitectura IA-32, se dispone de un espacio de direcciones adicional, exclusivo para la e/s, denominado espacio de direcciones de entrada/salida. Para que la CPU pueda acceder a dicho espacio, se usan instrucciones especiales: In y Out.

Hay periféricos que, además del espacio de direcciones exclusivo, usa direcciones de memoria, pues necesita mucha memoria para funcionar correctamente (como es el caso de la tarjeta de vídeo).

Protección del sistema de entrada/salida

Determina quién puede acceder al sistema de e/s. En el caso de los sistemas operativos multitarea, cualquier acceso al sistema de e/s se realiza por mediación del sistema mediante funciones especiales. La protección se implementa de la siguiente manera:

- Si los interfaces están ubicados en el **espacio de direcciones de memoria**, se asigna un nivel de acceso a las páginas de supervisor, para que solo puede acceder el sistema operativo.
- Si están ubicados en el **espacio de direcciones de entrada/salida**, el acceso se realiza con instrucciones especiales y el sistema operativo controla la ejecución de los mismos (controlando el campo IOPL del registro EFLAGS, por ejemplo).

Técnicas de entrada/salida

Dos problemas a resolver son:

- ¿Quién se encarga de mover la información?
- ¿Cómo se sincroniza la CPU con el dispositivo?

Para resolverlos se dispone de diferentes técnicas como:

- E/S programada con muestreo.
- E/S con interrupciones.
- DMA (direct memory access) o Acceso directo a memoria.
- Procesadores de e/s.

Las técnicas inferiores en la lista son más eficientes, es decir, consumen menos tiempo de CPU.

E/S programada con muestreo

Es la más simple e **ineficiente** de todas. Consiste en lo siguiente:

En todo interfaz existe un registro de control y/o de estado y un registro de datos. El registro de datos almacena la información que se va a intercambiar. El registro de estado indica cuando se lee, escribe o espera en el dispositivo.

La CPU comprueba constantemente dicho registro hasta que su estado es el esperado. Mientras que la CPU muestrea los registros del periférico no puede realizar otras tareas.

Una versión de este sistema es el muestreo periódico, que dedica un tiempo de espera a cada periférico. Para ello hay que garantizar que un programa se ejecute en un periodo fijo de forma precisa, lo cual es complicado.

E/S con interrupciones

Cuando la interfaz tiene algo que comunicar, 'avisa' a la CPU por medio de una interrupción. La CPU ejecuta instrucciones y, al finalizar cada una, comprueba las interrupciones. Si no hay ninguna, continúa ejecutando instrucciones con normalidad. En caso contrario, pasa a ejecutar un programa denominado **rutina de tratamiento de la interrupción** o **manejador de interrupción**.

Utilizando este sistema surgen tres problemas fundamentales:

1. **Priorización:** ¿Qué sucede en caso de que surjan simultáneamente dos interrupciones de distintos dispositivos?. Se consideran simultáneas cuando la CPU, al comprobar las interrupciones, detecta dos activadas (no tiene por qué ser simultáneamente en el tiempo).
2. **Posibilidad de inhabilitar interrupciones:** Ha de ser posible deshabilitar la 'escucha' de interrupciones en momentos críticos (por ejemplo en un cambio en la tabla de páginas). Igualmente deberían de poderse rehabilitar. En la arquitectura IA-32, esto se implementa mediante un bit en el registro EFLAGS.
3. **Identificación de la interfaz:** Se debe diferenciar el periférico que generó la interrupción. De ello se ocupa el **controlador de interrupciones**.

Si una interrupción es muy prolongada, sigue apropiándose de la CPU, evitando la ejecución de otras tareas.

DMA (Direct Memory Access) o Acceso Directo a Memoria

En este caso, es el propio periférico el que se encarga de mover los datos, dejando libre la CPU. Para ello, la CPU **le cede al periférico el control de los buses**.

¿Cuándo se usa DMA?

Si se dispone de un sistema operativo multitarea y cuando el controlador DMA es capaz de mover los datos más rápido que la CPU.

El DMA clásico (obsoleto)

El dispositivo **no es inteligente**, es decir, no ejecuta instrucciones. Es un dispositivo compartido para todos los interfaces de periféricos del sistema. Tiene el siguiente funcionamiento:

1. La CPU programa el controlador DMA con la operación a realizar. La CPU señala la cantidad de información y las direcciones de origen y destino.
2. La CPU, a través del sistema operativo, programa la interfaz para trabajar con DMA.
3. La interfaz, cuando está preparada, avisa al controlador DMA para que comience la transferencia. En ese momento, el controlador toma el control de los buses y realiza la transferencia.
4. Cuando el controlador DMA finaliza, avisa a la CPU mediante una interrupción.

Bus mastering (DMA moderno)

Cada interfaz tiene un controlador que es capaz de actuar como maestro del bus. Esto le permite usar los buses como si fuera la CPU. Estos controladores son dispositivos con cierta 'inteligencia', es decir, pueden transferir la información sin necesidad de que intervenga la CPU. Cuando el proceso del controlador termina, se lanza una interrupción.

Procesadores de entrada/salida

Se trata de interfaces con capacidad de **bus mastering** que incluyen una CPU con elevada capacidad de proceso. En ocasiones, es más compleja que la CPU principal.

La característica principal de estos procesadores es su inteligencia, con lo que pueden ejecutar órdenes de alto nivel. Si contamos los procesadores de entrada/salida más potentes, la mayoría de ordenadores personales pueden ser considerados como multiprocesador (aunque no son procesadores genéricos y, por lo tanto, no pueden realizar las mismas tareas que el procesador principal).

Interrupciones en la arquitectura IA-32

Las interrupciones se usan en periféricos con una tasa de entrada no muy elevada (teclado, ratón, puerto serie, etc).

La línea de interrupción a la CPU se denomina INT R (interrupt request). La CPU acepta, o no, las interrupciones según el bit IF del registro EFLAGS o FLAGS. Si el bit está a 1, las acepta. Si está a 0, las rechaza.

La CPU acepta las interrupciones entre el fin de una instrucción y el comienzo de la siguiente. En este momento se producen dos ciclos especiales denominados ciclos de reconocimiento de la interrupción. Durante estos ciclos sucede lo siguiente:

- Se notifica la aceptación de una interrupción (se activa la línea IACK),

- Se leen de los 8 bits bajos del bus de datos un número que identifica al elemento que ha pedido la interrupción. El número obtenido se denomina número de vector de interrupción.

Nota: Con 8 bits podemos formar 256 combinaciones.

Este número actúa como índice en la IDT para encontrar la dirección donde comienza la rutina de tratamiento de la interrupción.

Nota: la IDT es la tabla de descriptores de interrupción, similar a la GDT o la IDT.

Un tipo especial de interrupciones son las **interrupciones no enmascarables**, que son atendidas siempre y se generan ante fallos graves del sistema en la entrada NMI.

APIC (Advanced Programmable Interrupt Controller)

Se denomina PIC a los procesadores de control de interrupciones en general. El APIC está pensado para trabajar con sistemas multiprocesador. Todos los APIC se unen mediante un bus que termina en un APIC de entrada/salida, que gestiona a todos los demás. El APIC de entrada/salida tiene, al menos, 24 líneas de interrupción. Asociado a cada línea hay un registro en el que se escribirá el número de vector de interrupción del dispositivo.

Ejercicio:

En un computador en el cual hay un procesador con arquitectura IA-32 y un sistema operativo Windows ejecutándose en modo de memoria plano, se extrae la siguiente información de la IDT:

Número	Atributos (16 bits)	Selector (16 bits)	Desplazamiento (32 bits)
47	XXXX	0008	B8451960
48	XXXX	0008	C12F4620
49	XXXX	0008	B9805240
50	XXXX	0008	C1452470

IDT (Leyenda)	
Registro	Estado
Bit 0	CF
Bit 6	3F
Bit 7	5F
Bit 8	7F
Bit 9	IF
Bit 11	OF
Bits 12 y 13	IOPL

En este computador descrito, en un momento dado se está ejecutando la primera del siguiente grupo de instrucciones:

Dirección	Instrucción
0040100C	mov bl, [csi]
0040100E	add al, bl
00401010	inc esi

Mientras se ejecuta dicha instrucción se produce la interrupción 2Fh y, al concluir la instrucción, el registro EFLAGS es XXXX0282h

a) ¿Cuál será la siguiente dirección virtual a la que accederá la CPU del computador tras la ejecución de la instrucción en curso?

Expresamos el registro EFLAGS en binario:

XXXX 0000 0010 1000 0010

Vemos que el bit 9 (IF) está activado, luego se aceptan las interrupciones.

La interrupción 2Fh en decimal es la 47.

En un modelo de memoria plano, los registros de la GDT están a 0, por lo tanto la dirección que se ejecuta es, directamente, B8451960.

b) Si el grupo de instrucciones forma parte de un búfer, y en una iteración posterior, ejecutando la misma instrucción, se genera la interrupción 31h, sabiendo que al concluir la misma el registro EFLAGS tiene el valor XXXX0182h ¿Cuál será la siguiente dirección virtual a la que accederá el computador tras la ejecución de la instrucción en curso?

En este caso, al convertir el valor del registro EFLAGS a binario, vemos que el bit IF está a cero:

```
XXXX 0000 0001 1000 0010
```

Por lo que no acepta interrupciones y la ejecución de instrucciones no ve interrumpido su curso. La siguiente dirección que se consulta es la de la siguiente instrucción: 0040100E

El sistema de interconexión

Topologías de interconexión

Hay diversas alternativas para conectar componentes:

- **Conexión punto a punto (canal dedicado):** existe siempre una conexión directa entre dos dispositivos cualquiera. Como desventaja, nos encontramos con muchas conexiones, lo que con un número elevado de dispositivos, complica la estructura.
- **Conexión en estrella:** reduce el número de conexiones aunque no las optimiza. El concentrador actúa como un cuello de botella.
- **Conexión en bus:** todos los dispositivos acceden a la información en bus. Solo un dato puede viajar a la vez.

Nota: El bus AGP no es un bus como tal, si no un canal punto a punto que conecta la memoria a un dispositivo.

Los buses

Un bus es un **canal único y compartido de intercambio de información**. También es un conjunto de líneas al que se conectan los dispositivos y en el cual éstos escriben información que puede ser leída por todos los demás.

El dispositivo que escribe en el bus es denominado **maestro del bus**.

En un bus hay distintos tipos de líneas:

- **Líneas de datos:** se utilizan para transferir la información.
- **Líneas de direcciones:** se utilizan para indicar la dirección sobre la que actuar.
- **Líneas de control:** se utilizan para indicar la operación a realizar y para sincronizar algunos dispositivos.

- **Líneas de alimentación:** algunos buses llevan líneas para alimentar los dispositivos que están conectados a ellos (por ejemplo, el bus USB).

Concepto de ciclo de bus

Para que los periféricos puedan utilizar el bus, se establecen una serie de reglas. Estas reglas son los **protocolos de comunicaciones**.

El ciclo de bus se define como cada una de las operaciones que se pueden llevar a cabo en dicho bus, y viene especificado por una secuencia de bits determinada en las líneas del bus.

Para establecer esa secuencia se siguen los siguientes pasos:

1. El maestro del bus coloca en las líneas de direcciones del bus la identificación de la posición del dispositivo con el que vamos a comunicarnos.
2. Se activa la línea de control correspondiente (lectura, escritura, etc).
3. El dispositivo responde trasladando la información a través de las líneas del bus.

Un **ciclo de lectura** ocurre cuando el maestro del bus lee un dato de cierto dispositivo siguiendo estos pasos:

1. El maestro del bus coloca en las líneas de direcciones la identificación de la posición del dispositivo a leer.
2. El maestro del bus activa la línea de lectura.
3. El dispositivo responde situando el dato en el bus.

Un **ciclo de escritura** ocurre cuando el maestro del bus escribe un dato en cierto dispositivo siguiendo estos pasos:

1. El maestro del bus coloca en las líneas de direcciones del bus la identificación de la posición del dispositivo con el que vamos a comunicarnos.
2. El maestro del bus coloca en las líneas de datos el dato que queremos escribir.
3. Se activa la línea de escritura.

Otros ciclos son:

- Ciclo de reconocimiento de interrupción.
- Ciclo de configuración.
- Ciclos especiales para transferencias por bloque.

Características de un bus

- **Sincronización:** entre sus líneas de control, se utiliza una señal de reloj que permite secuenciar las etapas del ciclo del bus. Si un bus es síncrono, es más simple de realizar y más rápido que uno asíncrono. Los buses asíncronos secuencian la información mediante líneas de control adicionales (línea R, de Request, para peticiones y línea Ack, de Acknowledge, para reconocimiento).
- **Tamaño de los datos:** el número de bits con los que trabaja el bus, es decir, el número de líneas de datos.
- **Serie o paralelo:** el bus serie envía los datos de forma secuencial, mientras que el bus paralelo envía los datos simultáneamente (número de líneas = número de bits del dato). Si un bus es serie, lo habitual es que disponga de una línea de datos.
- **Multiplexado o no multiplexado:** si es multiplexado, algunas líneas podrán usarse para más de una tarea. Lo habitual es multiplexar líneas de direcciones y de datos.

- **Ancho de banda teórico:** velocidad de transmisión máxima en condiciones ideales.
- **Longitud máxima del bus:** los buses muy largos dan problemas. En el caso de los buses síncronos, la señal de reloj se retarda con tamaños de bus grandes. Además los buses de gran tamaño actúan como campos electromagnéticos.
- **Plug and Play:** capacidad de reconocer y configurar automáticamente un dispositivo que se conecte al bus.
- **Conexión en caliente:** los dispositivos se pueden conectar sin necesidad de desconectar el equipo.

Bus con múltiples maestros: concepto de arbitraje

Actualmente, muchos dispositivos tienen capacidad para controlar el bus. Para evitar el caos en el bus, es necesario un orden determinado.

Debemos tener en cuenta:

- **Priorización:** ante peticiones simultáneas de uso del bus, se concede al que tenga mayor prioridad.
- **Imparcialidad:** todos los dispositivos que pueden usar el bus, pueden ser maestros del bus en alguna ocasión.

Hay varios tipos de arbitraje_

- **Centralizado:** se utiliza un dispositivo especial que administra el bus, denominado árbitro del bus. Todos los dispositivos con capacidad de ser maestros, están conectados al árbitro con unas líneas de petición del bus y de conexión al bus. El árbitro se encarga de activar las líneas de conexión al bus. Como inconveniente destacamos el uso obligatorio de un árbitro. Este sistema es usado en el bus PCI.
- **Distribuido por auto selección:** cada dispositivo que desea ser maestro, coloca un código que lo identifique en la línea de datos. Todos los dispositivos examinan el bus y saben cuando es su turno para usarlo. Este sistema es usado por los multiprocesadores Pentium.
- **Arbitraje distribuido por detección de colisión:** cuando no se usa el bus, se utiliza y si no detecta colisión, será el maestro del bus. Este sistema lo usa el bus Ethernet.

El bus PCI

Historia

El PC se plantea como un dispositivo ampliable y configurable. El primer bus de conexión de ampliaciones fue el **XT**. Con el aumento de velocidad del procesador, se creó el bus AT que tenía una velocidad de 8,33 Mhz.

Una versión mejorada del anterior, el bus **EISA**, no tuvo demasiado éxito y permitió la aparición de alternativas como el bus **VESA** que se basaba en conectar el bus directamente al procesador (trabajando a igual velocidad) pero que contaba con problemas de compatibilidad pues con la rápida evolución de los procesadores, éste bus debía sacar versiones acordes con éstos.

Intel desarrolló el bus **PCI** (Peripheral Component Interconnect). Es un canal de intercomunicación de gran velocidad e independiente del bus del procesador. Es **síncrono, paralelo, multiplexado** y tiene soporte para **Plug and Play**. La especificación más frecuente del bus PCI contempla 32 líneas y una velocidad de 33 Mhz.

Ejercicio:

Si tenemos un bus PCI de 32 líneas y 33,3 Mhz ¿Cuál es el ancho de banda? (suponemos que en cada ciclo se puede transmitir información)

$32/8 = 4$ bytes

$4 \times 33,3 = 133$ Megabytes/segundo

Características

En toda transferencia participa un **iniciador** (maestro del bus) y un **destinatario** (dispositivo con el que se realiza la comunicación).

Existen múltiples dispositivos capaces de ser maestros del bus, por lo que se utiliza un dispositivo adicional que actúa como árbitro del bus (**arbitraje centralizado**).

Todo maestro dispone de dos líneas:

- **REQ**: para solicitar el bus.
- **GNT**: para recibir el control del bus.

Al ser un bus **síncrono**, el funcionamiento está sincronizado con una señal de reloj (CLK).

Ciclo de lectura óptimo

1. El iniciador activa la línea FRAME para llevar a cabo una operación en el bus. Coloca en las líneas AD (direcciones y datos) la dirección de la que desea leer. En las líneas C/BE (comando) indica la operación que desea realizar.
2. El iniciador libera las líneas AD. En las líneas C/BE se indica el número de bytes que va a leer en cada ciclo (de 1 a 4). Por ejemplo, si se escribe 0000 se indica que se leerán 4 bytes.
3. El iniciador repite el mensaje de las líneas C/BE para los siguientes paquetes y el destinatario envía el primer paquete de datos.
4. El iniciador lee el primer dato de las líneas AD y utiliza las líneas C/BE para el siguiente dato. El destinatario ya colocó el segundo dato en las líneas AD.
5. En el penúltimo dato, el iniciador desactiva la línea FRAME y libera las líneas C/BE. El destinatario envía el último dato.

Ciclo de escritura óptimo

1. El iniciador activa la línea FRAME para indicar el inicio de una operación en el bus. Coloca en las 32 líneas AD, la dirección que identifica la posición donde quiere escribir. Coloca en las 4 líneas C/B el código de la operación de escritura.
2. El dispositivo destinatario está avisado de la operación de escritura. El iniciador coloca en las 4 líneas C/BE el número de bytes no válidos en el paquete enviado. El iniciador coloca en las líneas AD el primer dato a enviar al destinatario.
3. El destinatario lee del bus el primer dato. El iniciador coloca en la línea AD el segundo dato y en las líneas C/BE el número de bytes válidos.
4. Mientras el destinatario lee el penúltimo dato, el iniciador desactiva la línea FRAME para indicar el fin del uso del bus. Coloca en las líneas AD el último dato a enviar e indica, en las líneas C/BE, el número de datos válidos.

Conexión de los componentes en un PC moderno

Tenemos múltiples dispositivos conectados a un bus, lo que puede desembocar en un bus muy largo (con la correspondiente pérdida de sincronización e interferencias). Podemos establecer múltiples conexiones, con lo que el bus se convierte en un cuello de botella.

Aprovechando las diferencias de velocidad entre los dispositivos, podemos establecer una jerarquía de buses.

En la arquitectura IA-32 existen tres concentradores:

- **North bridge** (AGP, RAM).
- **South Bridge** (Ethernet, PCI, Discos, Controlador DMA y de interrupciones).
- **Super I/O** (PS/2, serie, infrarrojos, paralelo).

Al conjunto de North bridge y South bridge se le denomina **chipset**.

Periféricos

Introducción

Los periféricos han sido muy importantes en el desarrollo y expansión de los computadores, evolucionando al tiempo que ellos. Según la dirección en que viaja la información podemos establecer la siguiente clasificación:

- **Periféricos de entrada:** sirven para introducir información en el ordenador. Por ejemplo, teclado, ratón, escáner, joypad, joystick, lector de bandas magnéticas, reconocimiento de voz, etc.
- **Periféricos de salida:** sirven para presentar la información fuera del computador. Por ejemplo, monitor, impresora, plotter, etc.
- **Periféricos de entrada y salida:** sirven para introducir y presentar información del ordenador. Por ejemplo, modem, tarjeta de red, pantalla táctil, etc.
- **Periféricos de almacenamiento:** son un tipo especial de periféricos, pues el movimiento de información no tiene como finalidad la representación, sino su almacenamiento. Por ejemplo, disco duro, cd, dvd, pendrive, etc.

Periféricos de almacenamiento

Se busca un coste bajo y una elevada capacidad, persistencia de la información y facilidad de transporte.

Estos tipos de periféricos usan principalmente tres tipos de tecnologías:

- Almacenamiento magnético (cintas).
- Almacenamiento óptico (CD y DVD).
- Almacenamiento magneto/óptico (Minidisc).
- Almacenamiento basado en semiconductores (RAM, ROM, FLASH, pendrives).

Almacenamiento magnético

Se basan en el uso de propiedades magnéticas para almacenar información. En concreto se usan materiales **ferro-magnéticos**.

Almacenamiento de datos en materiales ferro-magnéticos

Existen campos de dominios magnéticos que se orientan con el campo magnético aplicado. El efecto magnético permanece aunque se retire el campo.

Relación entre magnetización y campo magnético aplicado

La magnetización aumenta al incrementar la fuerza del campo hasta que se alcanza el nivel de saturación magnético. Al disminuir el campo, la magnetización apenas disminuye. Cuando no hay campo aplicado, se mantiene un nivel de magnetización llamado **remanencia magnética**.

Si aplicamos un campo magnético negativo, disminuye la magnetización hasta eliminarla por completo. Si mantenemos el campo negativo, la magnetización satura de forma negativa. Este ciclo funciona de igual forma que en la parte positiva. Este proceso se denomina **ciclo de histéresis**.

Cada bit se representa por una zona magnetizada de forma **homogénea**. Las zonas que representen un 0 estarán polarizadas inversamente a las que representen un 1. La remanencia puede entenderse como la fuerza para conservar un determinado bit, es decir, la **volatilidad de la información**.

El coste del campo magnético para polarizar inversamente la magnetización se corresponde con la sobre escritura del bit.

Definimos la **densidad de almacenamiento** como el número de bits almacenados por área del dispositivo. Para aumentar dicha densidad hemos de reducir los dominios que se usan para almacenar un bit.

Grabación y lectura de datos usando materiales magnéticos

Vamos a estudiar las propiedades magnéticas del material para transformar datos a magnetizaciones (escritura) y magnetizaciones a datos (lectura). Para ello se utilizan los principios de la inducción electro-magnética.

Las cabezas de escritura son imanes cuya polaridad esta determinada por el sentido de la corriente que fluye a través de la bobina que contiene. De esta forma, al cambiar el sentido de la corriente, escribimos un 1 o un 0.

Por el contrario, las cabezas de lectura detectan los cambios de orientación magnética del material ferro-magnético. Los cambios se detectan en las **fronteras** de los dominios y ello nos permite reducir el tamaño de los mismos.

Codificación de la información en medios magnéticos

Hay muchas formas de polarizar la información: modulando la frecuencia, la amplitud, etc. Nosotros vamos a estudiar algunos tipos de **codificación modulada por frecuencia (FM)**.

Esta técnica utiliza una señal intermedia entre bits y polarizaciones. La frecuencia de la señal viene determinada por la secuencia de bits. Se presenta como un tren de pulsos. Cada pulso se entiende como un cambio en la polarización.

Código FM

- Hay un pulso en el centro de cada bit 1 (pulso de bit).
- Hay un pulso al comienzo de todos los bits (pulso de reloj).

Código MFM

- Hay un pulso en el centro de cada bit 1 (pulso de bit).

- Hay un pulso al comienzo de un bit si en el bit anterior NO hubo pulso de bit (bit 0) y en el actual no hay pulso de bit.

Código M²FM o MMFM

- Hay un pulso en el centro de cada bit 1 (pulso de bit).
- Hay un pulso al comienzo del bit si en el bit anterior no hubo pulso de bit (bit 0) ni de reloj y en el actual no hay pulso de bit (bit 0).

Ventajas e inconvenientes de los códigos anteriores

En el código FM, los pulsos de reloj no aportan información pero sirven para sincronizar la lectura/escritura.

Los códigos MFM y MMFM aprovechan mejor el material, pudiendo reducir el tiempo y espacio de la transmisión.

MFM utiliza aproximadamente la mitad de polarizaciones que el FM. MMFM evita alguna polarización del MFM.

Unidades de disco

Organización de la información en el disco

- Pistas (tracks), numeradas de 0 a N-1.
- Sectores, que almacenan 512 o 1024 bytes. Son la mínima unidad de información a la que se puede acceder.
- Cilindros, que son el conjunto de pistas con la misma posición de todos los discos.

Esta organización se define al realizar un formateo de bajo nivel. Los clusters están definidos por el sistema operativo como una agrupación de dos o más sectores (según el operativo) y es la mínima información que se puede manejar sobre el disco.

En un formateo a alto nivel se copia a la pista 0 de la unidad una estructura de datos denominada **FAT** (File Assignment Table o Tabla de Asignación de Ficheros). En la FAT se asocia el nombre del fichero con el clúster de comienzo del mismo. Al final de cada clúster, se indica el siguiente clúster del fichero.

Si el disco usa **densidad de almacenamiento variable**, es simple de implementar pero desaprovecha material ferro-magnético.

Si, por el contrario, la **densidad de almacenamiento** es fija, aprovecha mejor el material magnético pero requiere un control de velocidad en el disco.

Parámetros básicos de una unidad de disco

- Capacidad: número total de bytes que almacena.
- Tiempo medio de acceso: tiempo que se tarda en acceder a la información para lectura o escritura.
- Tiempo de latencia: tiempo de espera a que la cabeza se mueva a un sector determinado.
- Tiempo de transferencia: tiempo en trasladar la información a la CPU.
- Densidad de almacenamiento: cantidad de información por área del disco.
 - TPI: pistas por pulgada.
 - BPI: bits por pulgada.

Disco flexible

- Usa material flexible.
- Gira a 360 r.p.m.
- Está organizado en 80 pistas por superficie.
- 18 sectores por pista.
- 512 bytes por sector.

Gira tan despacio porque las cabezas tocan la superficie del disco. Si girara a más velocidad, desgastaría el material magnético.

Disco Zip

- Formato propietario de IOMEGA.
- Almacena unos 100 o 200 Mb.
- Está hecho de material flexible.
- Las cabezas son más pequeñas y tocan muy ligeramente el disco.

El menor tamaño de las cabezas permite una densidad de almacenamiento mayor. Usa **densidad de almacenamiento fija**.

Discos duros

Están contruidos con material **rígido** (generalmente aluminio) sobre el que se coloca el material **ferro-magnético**. Las cabezas no tocan el disco, pues el disco gira tan rápido que genera aire cuyo movimiento levanta las cabezas lo suficiente. Cuando deja de girar, las cabezas se detienen en un sector especial denominado sector de aparcamiento.

Almacenamiento óptico

Se basa en el uso de propiedades ópticas de los materiales para almacenar información.

Principios de funcionamiento

Utiliza la misma tecnología que los CDs de audio, con la diferencia que en los CD-ROM se introduce información adicional para detectar y corregir errores.

Constitución física

La base es un **sustrato plástico transparente**. Sobre ella se aplica una **capa metálica reflectante** a la que se impregna una capa de protección.

Los datos se almacenan en muescas (**pits**) en la capa metálica, dejando espacios entre ellas (**lands**). Solo consta de una única pista que transcurre en espiral del centro al borde del disco.

Para interpretar la información, se emite un haz láser y se detecta la luz reflejada o la ausencia de ella.

Según la codificación **ISO9660** (Audio CD) el 1 se representa con una transición de luz a oscuridad (o viceversa) y el 0 con la ausencia de cambios. Cada 8 bits válidos se almacenan en 17 bits. Después, esos 8 bits válidos se agrupan con otros hasta ocupar 2352 bytes, magnitud que se denomina sector.

Cada segundo, 75 bloques son leídos. Por ejemplo, un CD de 74 minutos:

$74 \times 60 \times 75 = 333.000$ bloques

$(2352 \times 333000) / 1024 / 1024 = 746'9$ Mbytes.

Si el CD es de datos, cada sector lo componen 2048 bytes y se emplean 304 bytes adicionales para información de detección de errores.

Tipos de discos

CD-ROM: solo lectura.

CD-R: CD grabable solo una vez.

CD-RW: CD regrabable, hasta 1000 veces.

DVD-ROM: solo lectura.

DVD-R: grabable solo una vez (Pioneer)

DVD+R: grabable solo una vez (DVD Alliance)

DVD-RW: DVD regrabable hasta 1000 veces (Pioneer)

DVD+RW: DVD regrabable hasta 1000 veces (DVD Alliance)

El CD

Emplea una luz láser de 780 nanómetros (infrarrojo). La velocidad de transferencia a 1x es de 75 bloques/segundo, es decir, $75 \times 2048 / 1024 = 150$ Kb/seg.

A una velocidad de 60x, la tasa de transferencia aumenta hasta 9 Mb/seg.

El DVD

El DVD almacena, en su versión simple, 4.7 Gb de datos. Esto lo consigue reduciendo el tamaño de los pits y los lands además de estrechar la espiral que recorre el disco. Esto requiere un **láser más preciso** (de 635 a 650 nanómetros).

El DVD puede albergar una **doble capa**, permitiendo almacenar 8.5 Gb. Para realizar esto se superpone a la capa metálica, otra de material transparente que permite grabar pits y lands. Para leer una u otra capa, se modifica la distancia focal del láser.

Usando **doble cara** se consiguen 9.4 Gb. Si además empleamos doble capa sobre esta cara, conseguimos 17 Gb.

En vez de usar 17 bits para codificar 8 bits reales (como sucede con los CDs), se emplean 16 bits.

La **velocidad de transferencia** de un DVD (1x) es de 1.3 Mb/seg independientemente del contenido del disco.

EL DVD-RAM

Es un formato específico para datos. En el momento actual no es compatible con los DVD-ROM. A modo de curiosidad, podemos destacar que, al igual que los disquetes, utiliza una **carcasa externa**.

CDs y DVDs grabables y regrabables

Grabables

Solo pueden ser escritos una vez. Están formados por un **polímero fotosensible** (un tinte) depositado en un surco en espiral a lo largo del sustrato plástico del disco.

Dicho tinte es reflectante en condiciones normales. Al incidir el láser sobre el tinte lo 'quema' haciendo que pierda las propiedades reflectantes.

Regrabables

Pueden ser escritos alrededor de 1000 veces, tras un proceso de borrado. Están formados por una **aleación metálica** (plata, indio, antimonio y telurio) de estructura cristalina. Al calentar la estructura (600 °C), se funde y deja de ser reflectante en ese punto.

Si la estructura se calienta a 200°C recupera la estructura cristalina. Esta operación es la que se denomina como 'borrado'.

Almacenamiento magnético-óptico

Su máximo exponente es el **Minidisc** de Sony. Tiene una capacidad de 9 Gb en discos de tamaño 5'25".

Funcionamiento

Están formados de un **material magnético** que no es sensible a los campos magnéticos a temperatura ambiente, pero se vuelve afectado por ellos a temperaturas de alrededor de 300 °C.

Proceso de escritura

Un láser calienta una zona magnética del material para que el campo magnético de la cabeza actúe en ese punto.

Proceso de lectura

Se utiliza un láser que emite luz polarizada y dependiendo del ángulo de polarización de la luz reflejada, obtendremos un 0 o un 1.

Almacenamiento basado en semiconductores

Discos de estado sólido

Son semiconductores que sustituyen a discos en ambientes agresivos. En un principio necesitaba alimentación, pero ahora utilizan sistemas EEPROM. Son relativamente rápidos (aunque no tanto como el disco duro).

Periféricos de entrada

Su objetivo es introducir información en el ordenador.

El teclado

Para detectar la pulsación de una tecla, en el ciclo de exploración, se coloca cada fila a 1 de forma sucesiva y se detectan las columnas que están activas.

El controlador de teclado asigna a cada tecla un código, denominado código de scan o **keycode**. Dichos códigos no pertenecen a la tabla ASCII.

Para solucionar la pulsación de teclas simultáneas se envía la última tecla pulsada (del conjunto de teclas pulsadas a la vez).

El código es enviado bit a bit (en serie) a través de la línea **KBDATA**. Se envía un bit en cada ciclo de reloj que refleja **KBCLOCK**.

El ratón

Es un elemento de señalización en pantalla.

Los **ratones mecánicos** emplean una rueda que mueve unos discos con ranuras que interrumpen un haz de luz. Midiendo la velocidad de las interrupciones de la luz sabemos el desplazamiento en cada eje. La velocidad puede ser positiva o negativa, según el sentido del desplazamiento.

En el ratón serie (puerto serie) el desplazamiento se envía en dos palabras de 16 bits. El desplazamiento se mide en **mickeys** (1 mickey = 0'13).

El driver de ratón

Con la información recibida, se encarga de actualizar el puntero en pantalla, detecta la secuencia 'doble click' en un margen de tiempo y detecta la secuencia 'arrastrar' en otro intervalo de tiempo.

Periféricos de salida

Monitores CRT (tubo de rayos catódicos)

Los monitores usan un **barrido completo** (barren todas las líneas de la pantalla), mientras que los televisores realizan un **barrido entrelazado** (barren alternativamente las filas pares o las impares).

Los monitores tienen dos señales de sincronización:

- **HSYNC** (sincronización horizontal): se activa cada vez que el haz llega al final de una línea. Determina la frecuencia del barrido horizontal (FBH).
- **VSYNC** (sincronización vertical): se activa cada vez que el haz llega al final de la pantalla. Determina la frecuencia del barrido vertical (FBV).

Monitores a color

La tarjeta gráfica controla qué punto de color se activa (de los tres posibles). Cada punto suele tener intensidad variable (para poder generar toda la gama de colores) y esta varía entre 0v a 0'7v generalmente.

La relación de tamaño en estos monitores sigue la relación:

$$\text{Ancho} / \text{Alto} = 1'3$$

Monitores LCD (liquid cristal display)

Las moléculas se orientan en torno al eje director (también llamado **eje óptico**) y los cristales giran para transformar el haz que los atraviesa. Los monitores LCD **no emiten luz**, pues simplemente la bloquean o la dejan pasar. Por ello se necesita una fuente de luz adicional (generalmente un **halógeno**).

Ejercicio 1.1:

1)

$512 * 18 * 80 * 2 = 1474560 \text{ bytes} \approx 1'44 \text{ Mbytes}$

2)

T.busqueda + T.Latencia + T.Transferencia

T.Busqueda = 100 msg

T.Latencia = $\frac{1}{2} * (360 \text{ r.p.m.} / 60)$

T.Transferencia = $\text{bytes} * (1 \text{ mseg} / (50 * 100 / 1000))$

83'3 mseg

3)

$(1 \text{ pulgada} / 135 \text{ pistas}) * (25'4 \text{ mm} / 1 \text{ pulgada}) = 0,188 \text{ mm/pista}$

Tamaño de la zona de pistas:

$80 \text{ pistas} * 0'188 = 15'04 \text{ mm}$

Radio medio de la pista:

Información en una pista =

$18 \text{ sectores} * (512 \text{ bytes} / \text{sector}) * (8 \text{ bits} / \text{byte}) = 73728$

También sirve así:

$\text{Capacidad total} / \text{n}^\circ \text{ pistas} = (1474560 * 8) / (80 * 2) = 73728$

Longitud de pista:

$\text{Bits en pista} / \text{bits por pulgada} = 73728 / 17434 = 4'229 \text{ pulgadas}$

El radio es $4'229 / 2'14 = 1'97 \text{ cm}$

Ejercicio 1.3:

1)

Nº sectores por pista:

$2 * \pi * 5 \text{ cm/pista} * 6518'99 \text{ bits/seg} = 204800 \text{ bits/pista}$

$(1024 + 256) \text{ bytes/sector} * 8 \text{ bits/byte} = 10240 \text{ bits/sector}$

$204800 / 10240 = 20 \text{ sectores/pista}$

Capacidad neta:

$1024 \text{ bytes/sector} * 20 \text{ sectores/pista} * 256 \text{ pistas/superficie} * 4 \text{ superficies} = 20 \text{ Gb}$

3)

$T.\text{Busqueda} = 3 \text{ mseg} + (51-1) * 0'2 \text{ mseg} = 13 \text{ mseg}$

$T.Latencia = 3000 \text{ rev/min} * (1 \text{ min} / 60 \text{ seg}) * (1 \text{ seg} / 1000 \text{ mseg}) = 1 \text{ sector} / \text{seg}$

Al final de la búsqueda nos situamos en el sector $5+13 = 18$

Para llegar al sector 15 tenemos que acabar una vuelta (20-18) y añadir algunos sectores más hasta situarnos correctamente:

$(20-18) + 15 = 17$ sectores que necesitan 17 mseg

$T.Transferencia = (1280 \text{ bytes/sector} * 8 \text{ bits/byte}) / 10240000 \text{ bits/seg} = 0'001 \text{ seg/sector}$

Impresoras

Las impresoras pueden estar orientadas a **carácter** o a **página**. Las que están orientadas a carácter admiten caracteres de control (también llamados secuencia de escape) para advertir tabulaciones, saltos de línea, etc.

Para describir las páginas que han de imprimir existen diversos lenguajes de descripción, como por ejemplo:

- **PS**: Adobe PostScript.
- **HP-PDL**: Hewlett-Packard page description language.
- **GDI**: Graphical Device Interface.

En el lenguaje **GDI** es el propio procesador de la CPU el que genera el mapa de puntos, por lo que es algo más lento que si existiera un procesador dedicado en la impresora, pero más económico.

Las unidades en que se mide la velocidad de impresión son:

- **CPS**: caracteres por segundo (para impresión orientada a carácter).
- **PPM**: páginas por minuto (para impresión orientada a página).

Impresoras de aguja

Un imán repele la aguja, que presiona una cinta entintada y marca la página. Tras esto el imán deja de actuar y el muelle hace retroceder a la aguja a su posición de origen.

Este tipo de impresoras son **orientadas a carácter**. Para mejorar la calidad se debe aumentar el número de agujas y hacerlas lo más cuadradas posibles (para reducir el espacio entre puntos).

Son útiles para manejar papel continuo y para hacer duplicados, calcando el documento original.

Impresoras de inyección de tinta

Las boquillas por las que se despiden las gotas de tinta se disponen de forma matricial. El número de boquillas varía de 28 a 128.

Desde cada boquilla se pueden disparar 12000 gotas por segundo. Cada gota es aproximadamente la diezmilésima parte de una gota de agua.

Impresoras láser

Son impresoras orientadas a la página. El funcionamiento de estas impresoras se reducen a los siguientes pasos:

1. Toda la información a imprimir se traduce a un mapa de puntos. El tambor comienza a girar.
2. El tambor tiene una base de aluminio y está recubierto por un material fotosensible. Cuando la luz incide sobre dicho material pierde su carga electrostática. Al tambor se le aplica una carga electrostática de -600V.
3. El tambor cargado alcanza la posición del láser. El haz láser describe líneas de puntos, de la matriz de puntos, sobre el tambor. El controlador del láser enciende el láser en las zonas de puntos y lo apaga cuando no hay puntos. El láser se refleja en un espejo que determina la dirección del láser, determinando el lugar de incidencia sobre el tambor. El material fotosensible pierde su carga en los puntos a imprimir.
4. El tambor modificado llega al aplicador del tóner. El tóner es una tinta seca formada por un fino polvo de resina plástica, compuestos orgánicos y partículas de hierro. El tóner está cargado con el mismo signo que el tambor. La tinta es atraída por las zonas del tambor en que incidió el láser (pues fueron descargadas).
5. Paralelamente, el papel se carga electrostáticamente con una carga positiva.
6. Al entrar en contacto con el tambor, el papel atrae las partículas de tinta.
7. Se elimina la carga del papel y se imprimen las partículas de tinta en el papel mediante los rodillos de fricción, que aplican calor (fundiendo la resina plástica) y presionan la tinta fijándola al papel.
8. Por último se limpia el tambor de los residuos de tóner, antes de comenzar el ciclo de nuevo.

Ejercicio 2.1:

1)

Traducimos los caracteres por las teclas pulsadas. Hay caracteres que no vienen en el teclado, pero que difieren en 80 posiciones con alguno existente. Estos caracteres significan que el usuario ha levantado la tecla. Por ejemplo:

2A: pulsar Mayúsculas.

AA: soltar Mayúsculas.

Los códigos serían los siguientes:

Mayúsc↓, H↓, H↑, Mayúsc↑, O↓, O↑, K↓, K↑, Backspace↓, Backspace↑, L↓, L↑, A↓, A↑, Enter↓, Enter↑

2)

El mensaje resultante es "Hola".