

Subsistema de Entrada/Salida

Organización de computadoras

Universidad Nacional de Quilmes

<http://orga.blog.unq.edu.ar>

Repaso

Memoria Caché

- Principios de localidad
- Función de correspondencia
- Políticas de reemplazo
- Métodos de escritura
- Tiempo medio de acceso

Repaso

Memoria Caché

- Principios de localidad
- Función de correspondencia
- Políticas de reemplazo
- Métodos de escritura
- **Tiempo medio de acceso**

Tiempo medio de acceso

Tot Tiempo
total

tpoA tiempo
de un
acierto

tpoF tiempo
de un
fallo

$$\frac{Tot}{\#refs} = \frac{\#aciertos * tpoA + \#fallos * tpoF}{\#refs} =$$

Tiempo medio de acceso

TAC tiempo
de acceso
a caché

TAM tiempo
de acceso
a
memoria

$$\frac{Tot}{\#refs} = \frac{\#aciertos * tpoA + \#fallos * tpoF}{\#refs} =$$
$$= \frac{\#aciertos * TAC + \#fallos * (TAM + TAC)}{\#refs}$$

Tiempo medio de acceso

TAC tiempo
de acceso
a caché

TAM tiempo
de acceso
a
memoria

tA tasa de
aciertos

tF tasa de
fallos

$$\begin{aligned}\frac{Tot}{\#refs} &= \frac{\#aciertos * tpoA + \#fallos * tpoF}{\#refs} = \\ &= \frac{\#aciertos * TAC + \#fallos * (TAM + TAC)}{\#refs} \\ &= \frac{\#refs * tA * TAC + \#refs * tF * (TAM + TAC)}{\#refs}\end{aligned}$$

Tiempo medio de acceso

TAC tiempo de acceso a caché

TAM tiempo de acceso a memoria

tA tasa de aciertos

tF tasa de fallos

$$\begin{aligned} \frac{Tot}{\#refs} &= \frac{\#aciertos * tpoA + \#fallos * tpoF}{\#refs} = \\ &= \frac{\#aciertos * TAC + \#fallos * (TAM + TAC)}{\#refs} \\ &= \frac{\#refs * tA * TAC + \#refs * tF * (TAM + TAC)}{\#refs} \\ &= \frac{\#refs * tA * TAC}{\#refs} + \frac{\#refs * tF * (TAM + TAC)}{\#refs} \end{aligned}$$

Tiempo medio de acceso

TAC tiempo
de acceso
a caché

TAM tiempo
de acceso
a
memoria

tA tasa de
aciertos

tF tasa de
fallos

$$\frac{Tot}{\#refs} = \frac{\#aciertos * tpoA + \#fallos * tpoF}{\#refs} =$$

$$= \frac{\#aciertos * TAC + \#fallos * (TAM + TAC)}{\#refs}$$

$$= \frac{\#refs * tA * TAC + \#refs * tF * (TAM + TAC)}{\#refs}$$

$$= \frac{\#refs * tA * TAC}{\#refs} + \frac{\#refs * tF * (TAM + TAC)}{\#refs}$$

$$= tA * TAC + tF * (TAM + TAC) = (1 - tF) * TAC + tF * (TAM + TAC)$$

Tiempo medio de acceso

TAC tiempo
de acceso
a caché

TAM tiempo
de acceso
a
memoria

tA tasa de
aciertos

tF tasa de
fallos

$$\frac{Tot}{\#refs} = \frac{\#aciertos * tpoA + \#fallos * tpoF}{\#refs} =$$

$$= \frac{\#aciertos * TAC + \#fallos * (TAM + TAC)}{\#refs}$$

$$= \frac{\#refs * tA * TAC + \#refs * tF * (TAM + TAC)}{\#refs}$$

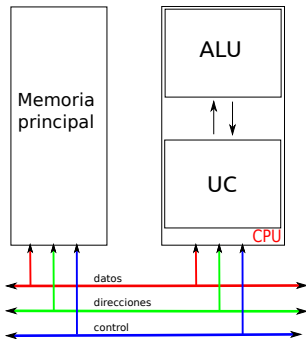
$$= \frac{\#refs * tA * TAC}{\#refs} + \frac{\#refs * tF * (TAM + TAC)}{\#refs}$$

$$= tA * TAC + tF * (TAM + TAC) = (1 - tF) * TAC + tF * (TAM + TAC)$$

$$= TAC - tF * TAC + tF * TAM + tF * TAC = TAC + tF * TAM$$

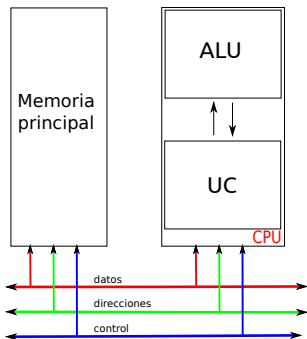
Módulo de Entrada/Salida

Módulo de Entrada/Salida



ALU + UC = neuronas
asociativas

Módulo de Entrada/Salida

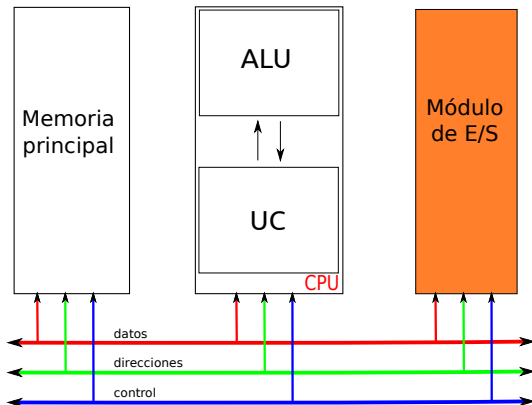


ALU + UC = neuronas
asociativas

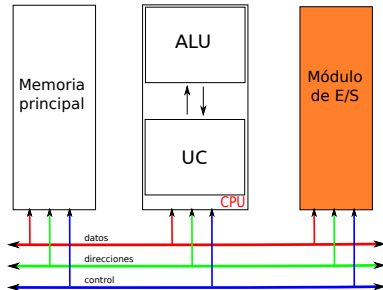


Hacen falta neuronas
sensoriales y motoras...

Arquitectura de Von Neumann



Módulo de Entrada/Salida



Objetivo: Permitir la interacción con dispositivos externos



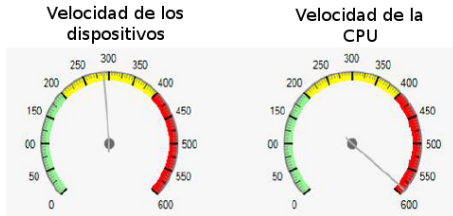
¿Porqué se necesita un módulo o interfaz?

¿Porqué necesitamos un módulo de Entrada/Salida?

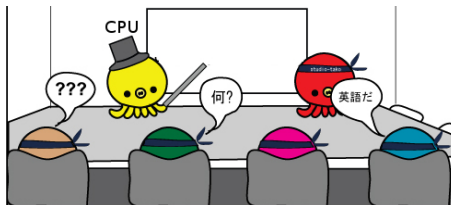
- 1 Distintos periféricos que se operan de diferentes formas: La cpu no puede estar preparada para manejarlos a todos.



- 1 Distintos periféricos que se operan de diferentes formas: La cpu no puede estar preparada para manejarlos a todos.
- 2 Velocidad de transferencia de datos de E/S diferente a la de la CPU o memoria



- 1 Distintos periféricos que se operan de diferentes formas: La cpu no puede estar preparada para manejarlos a todos.
- 2 Velocidad de transferencia de datos de E/S diferente a la de la CPU o memoria
- 3 Los periféricos usan formatos de datos diferentes a los de la CPU.



- 1 Distintos periféricos que se operan de diferentes formas: La cpu no puede estar preparada para manejarlos a todos.
- 2 Velocidad de transferencia de datos de E/S diferente a la de la CPU o memoria
- 3 Los periféricos usan formatos de datos diferentes a los de la CPU.
- 4 Calidad de datos sospechosa: ruido en la señal, defectos en la superficie de un disco



- 1 Distintos periféricos que se operan de diferentes formas: La cpu no puede estar preparada para manejarlos a todos.
- 2 Velocidad de transferencia de datos de E/S diferente a la de la CPU o memoria
- 3 Los periféricos usan formatos de datos diferentes a los de la CPU.
- 4 Calidad de datos sospechosa: ruido en la señal, defectos en la superficie de un disco
- 5 Transferencia interrumpida: error del dispositivo o condiciones especiales (ej: impresora sin papel)



¿Que debe hacer el módulo?

Funciones del módulo

Control y temporización

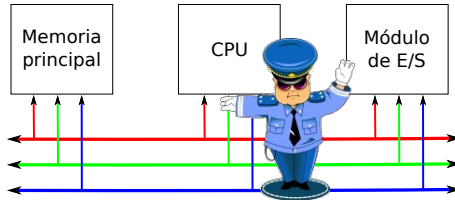
Comunicación con la CPU

Comunicación con el dispositivo

Buffering de datos

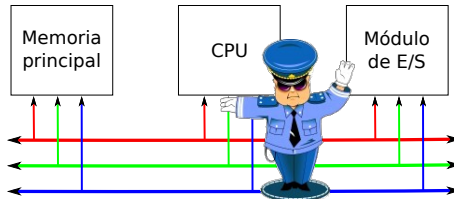
Detección de errores

Se deben compartir recursos:



- Bus del sistema
- Memoria Principal

Se deben compartir recursos:



- Bus del sistema
- Memoria Principal

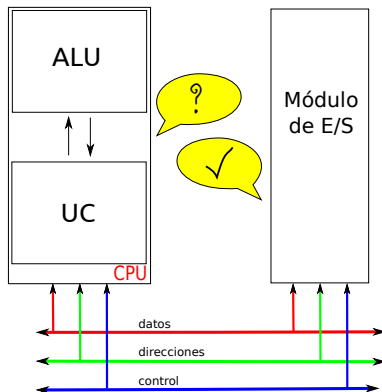
Se necesita coordinar el flujo de los datos: cada comunicación CPU-Modulo requiere una etapa de arbitraje del bus

Control y temporización

Ejemplo de una transferencia Modulo E/S $\rightarrow$ CPU

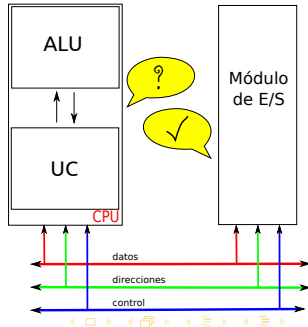
- 1 CPU pide al modulo el estado del dispositivo
- 2 El modulo informa el estado
- 3 Si el dispositivo está operativo y listo para transmitir, la CPU indica la transferencia de datos mediante un comando al modulo
- 4 El modulo obtiene una unidad de datos del dispositivo
- 5 Los datos se transmiten a la CPU

Comunicación con la CPU

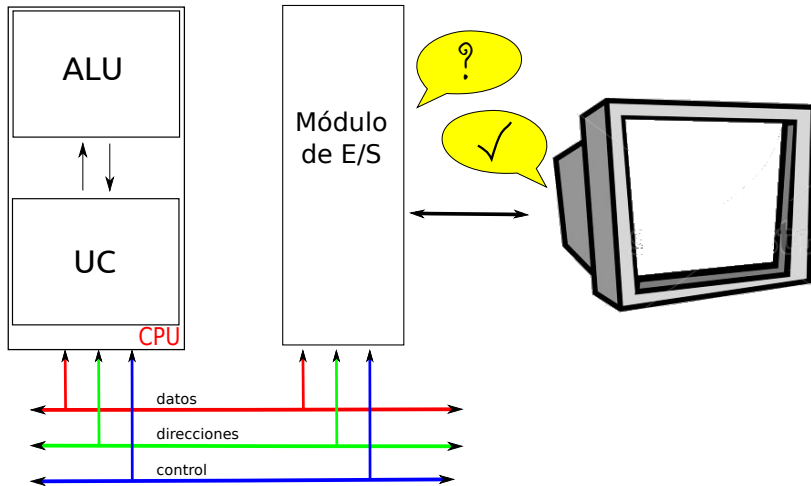


Comunicación con la CPU

- Interpretar de comandos: se utilizan líneas del **bus de control**.
- Reconocer dirección: poder distinguir los periféricos que controla mediante su dirección
- Reporte de estado: Mediante líneas de estado (Libre/Ocupado) o de condiciones de error
- Transferir datos: usando el bus de datos



Comunicación con el dispositivo

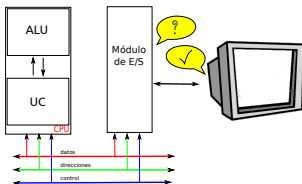


Comunicación con el dispositivo

Esto es muy específico para cada tipo de dispositivo.

En general incluye:

- Traducir las órdenes de la CPU en comandos específicos del dispositivo
- Pedir información de estado al dispositivo (libre/ocupado)
- Adaptar los datos que se envían desde/hacia la CPU



Buffering de datos



Buffering de datos



Buffering de datos

Existe una brecha entre tasas de transferencia y velocidades

- La CPU envía datos a alta velocidad, pero el dispositivo los toma mas lentamente.
- Los datos entrantes se empaquetan para no atar a la CPU a muchas operaciones de transferencia lentas (y que requiere el bus).

El módulo **almacena datos temporalmente** (*buffering* de datos)



Detección de errores

Existen dos tipos de errores que el módulo debe manejar:

- de mal funcionamiento del dispositivo
- de datos: durante la comunicación se cambia la cadena de bits.

Detección de errores

Existen dos tipos de errores que el módulo debe manejar:

- de mal funcionamiento del dispositivo
- de datos: durante la comunicación se cambia la cadena de bits.



Un método muy usado es el **cálculo de paridad** en cada caracter de información. El valor de este bit debe ser tal que en total se tenga una cantidad par (even parity) o impar (odd parity)

Estructura genérica de un módulo de E/S

Estructura de un módulo de Entrada/Salida

Ojo!

Los dispositivos pueden iniciar tareas por afuera del sistema, pero no transferencias hacia la CPU

Desde el punto de vista de los programas...

Instrucciones

La arquitectura debe proveer instrucciones (visibles al programador)

- Para leer un puerto
- Para escribir un puerto

Instrucciones

La arquitectura debe proveer instrucciones (visibles al programador)

- Para leer un puerto
- Para escribir un puerto

Dos posibles métodos de direccionamiento:

Mapeado a memoria Una parte del espacio de direccionamiento se reserva para dispositivos (y no para memoria).

Ej: `CMP [0080h], 5`

Aislado Se utilizan instrucciones especiales (IN y OUT) con un espacio de puertos fuera del espacio de direccionamiento a memoria.

Ej: `IN R0, 0080h`

`CMP R0, 5`

Instrucciones

Para ejecutar una instrucción de ES, la CPU provee comandos al módulo.

- **control:** Para activar el periférico.
- **test:** Para chequear condiciones de estado (disponible o encendido)
- **lectura:** Para obtener una unidad de información y ubicarla en el reg de datos (buffer)
- **escritura:** Para tomar el contenido del registro de datos y transmitirlo al periférico.

¿Cómo se trabaja con un módulo de Entrada/Salida?

Mecanismos de Operación de Entrada/Salida

Existen varias formas de realizar la gestión de las operaciones de entrada/salida a un dispositivo.

- 1 Entrada/Salida por **encuesta** (polling) o programada
- 2 Entrada/Salida por **interrupciones**
- 3 Entrada/Salida por **acceso directo a memoria**

E/S mediante encuestas(polling)

La CPU sondea periódicamente al módulo para conocer su estado, mediante la lectura de un registro de estado.

- 1 Cuando la CPU emite un comando de E/S debe esperar a que la operación termine.
- 2 El modulo realiza la operación y al final configura los bits del registro de estado.
- 3 La CPU es responsable de enterarse cuando un dispositivo necesita atención, enviarle datos y recibir datos de él.

Si la CPU no pregunta, no sabe si algún dispositivo necesita atención. El modo de programación casi siempre es con *busy-wait*

Ejemplo: Máquina de escribir

Implementar una máquina de escribir

Se leen teclas presionadas en un teclado y se imprimen en una impresora, asumiendo un modelo muy simplificado de ambos dispositivos

Teclado

Pone a disposición de la CPU los códigos de las teclas presionadas.

Si no se presiona ninguna tecla reportará código 0.

Se comunica a través del puerto 40A0

Impresora

Se comunica con la CPU a través del puerto 40B0, donde se obtiene el estado y se le da el código del carácter a imprimir.

Ejemplo: Máquina de escribir

Puerto	Sentido de la com	Significado
40A0h	Teclado → CPU	Código de la tecla presionada
40B0h	CPU → Impresora	Código a imprimir
40B0h	Impresora → CPU	Estado de la impresora: 1 En espera 2 Imprimiendo 3 Fuera de línea 4 Falta papel

Ejemplo: Máquina de escribir

Puerto	Sentido de la com	Significado
40A0h	Teclado → CPU	Código de la tecla presionada
40B0h	CPU → Impresora	Código a imprimir
40B0h	Impresora → CPU	Estado de la impresora: 1 En espera 2 Imprimiendo 3 Fuera de línea 4 Falta papel



¿Cómo es el programa?

Ejemplo: Máquina de escribir

ProgramaPrincipal: repetir

LeerTecla

EnviarImpresora

Ejemplo: Máquina de escribir

ProgramaPrincipal: repetir

LeerTecla

EnviarImpresora



ProgramaPrincipal: repetir

LeerTecla:

Si el código leído en 40A0 no es 0, leer código

EnviarImpresora:

Leer estado en el puerto 40B0

Si la impresora está disponible: enviar código por el puerto 40B0

Si no está disponible: volver a empezar

Ejemplo: Máquina de escribir

```
LeerTecla: IN R0, 0x40A0  
CMP R0, 0x0000  
JE LeerTecla  
RET
```

```
EnviarImpresora:  
IN R1, 0x40B0  
CMP R1, 0x0001  
JNE EnviarImpresora  
OUT 0x40B0, R0  
RET
```

```
ProgramaPrincipal:  
call LeerTecla  
call EnviarImpresora  
JMP ProgramaPrincipal
```

E/S mediante interrupciones

Problemas con la E/S programada:

- mucho tiempo de CPU está perdido en el busy-wait.
- la CPU podría invertir el tiempo en la ejecución de programas

Una alternativa es que la CPU continúe realizando algún trabajo útil luego de enviar una orden de E/S y el módulo de E/S lo interrumpa cuando esté preparado para intercambiar datos.

Interrupciones

El sistema debe proveer un mecanismo para interrumpir a la cpu.
¿Cuándo?

Interrupciones

El sistema debe proveer un mecanismo para interrumpir a la cpu.
¿Cuándo? $\Rightarrow$ Durante operaciones de Entrada/Salida

Interrupciones

El sistema debe proveer un mecanismo para interrumpir a la cpu.
¿Cuándo? $\Rightarrow$ Durante operaciones de Entrada/Salida

Pero también:

- En condiciones especiales como resultado de la ejecución de una instrucción: desbordamiento aritmético, división por cero, acceso a espacio de memoria no permitido
- Fallo de Hardware: error de datos, falta de energía, etc.
- Temporización: Para que el sistema operativo pueda ejecutar ciertas operaciones de manera regular.

Interrupciones: Mecanismo

- 1 El dispositivo envía una señal de petición de interrupción a la cpu.

Interrupciones: Mecanismo

- 1 El dispositivo envía una señal de petición de interrupción a la cpu.
- 2 La CPU suspende la ejecución del programa actual y salta a un programa de **gestión de interrupción**.

Interrupciones: Mecanismo

- 1 El dispositivo envía una señal de petición de interrupción a la cpu.
- 2 La CPU suspende la ejecución del programa actual y salta a un programa de **gestión de interrupción**.
- 3 Luego de dar servicio al dispositivo, la CPU continúa con el programa

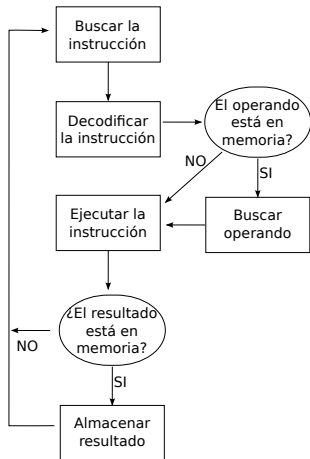
Interrupciones: Mecanismo

- El programa de usuario no debe incluir ningún código especial para considerar interrupciones.

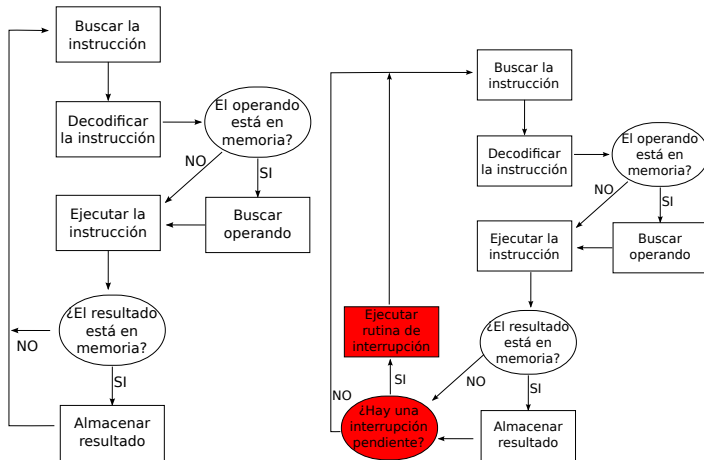
Interrupciones: Mecanismo

- El programa de usuario no debe incluir ningún código especial para considerar interrupciones.
- Implementación:
 - Se añade un subciclo de interrupción al ciclo de ejecución de instrucciones para comprobar la presencia de interrupciones.
 - En caso afirmativo:
 - 1 Se guarda el contexto del programa en curso: registros y otra información importante en una pila del sistema
 - 2 PC se carga con la dirección de la rutina de gestión de interrupción.

Interrupciones: Ciclo de ejecución de instrucción



Interrupciones: Ciclo de ejecución de instrucción



Entrada/Salida por DMA

Problemas de la E/S con interrupciones:

- es más eficiente que la E/S programada, pero requiere la intervención del CPU para transferir datos entre la memoria y el módulo de E/S.
- si se trabaja con dispositivos más veloces $\Rightarrow$ interrupciones muy seguidas y desperdicio de tiempo de CPU.

Entrada/Salida por DMA

Problemas de la E/S con interrupciones:

- es más eficiente que la E/S programada, pero requiere la intervención del CPU para transferir datos entre la memoria y el módulo de E/S.
- si se trabaja con dispositivos más veloces $\Rightarrow$ interrupciones muy seguidas y desperdicio de tiempo de CPU.



DMA (*Direct Memory Access*)

DMA (*Direct Memory Access*)

- En el caso del DMA, el dispositivo puede operar directamente sobre la memoria, siempre que la CPU le haya concedido el permiso para hacerlo.
- De esta manera, solo se avisa a la CPU al comienzo o al final de una operación sobre memoria y se consigue una tasa de transferencia de datos superior a la de los otros métodos.

Memoria Secundaria: Discos rígidos

Discos Rígidos

En cuanto al subsistema de memoria:

- No volátil
- Almacenamiento secundario
- De lectura y escritura
- Acceso Directo

Discos Rígidos

En cuanto al subsistema de memoria:

- No volátil
- Almacenamiento secundario
- De lectura y escritura
- Acceso Directo

Discos Rígidos

En cuanto al subsistema de memoria:

- No volátil
- Almacenamiento secundario
- De lectura y escritura
- Acceso Directo

Discos Rígidos

En cuanto al subsistema de memoria:

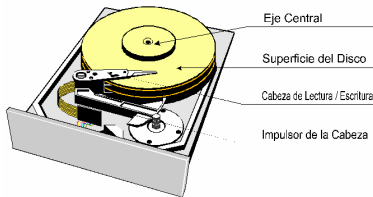
- No volátil
- Almacenamiento secundario
- De lectura y escritura
- [Acceso Directo](#)

Discos Rígidos

- Son platos de material magnético
- Para las operaciones de lectura y escritura, el cabezal permanece quieto mientras el disco gira debajo
- El plato está organizado en anillos concéntricos denominados pistas
- Todas las pistas tienen la misma capacidad, para simplificar la electrónica. Por lo tanto la densidad aumenta en las pistas mas centrales

Discos Rígidos

- Son platos de material magnético
- Para las operaciones de lectura y escritura, el cabezal permanece quieto mientras el disco gira debajo



- El plato está organizado en anillos concéntricos denominados pistas
- Todas las pistas tienen la misma capacidad, para simplificar la electrónica. Por lo tanto la densidad aumenta en las pistas mas centrales

Discos Rígidos

- Son platos de material magnético
- Para las operaciones de lectura y escritura, el cabezal permanece quieto mientras el disco gira debajo
- El plato está organizado en anillos concéntricos denominados pistas
- Todas las pistas tienen la misma capacidad, para simplificar la electrónica. Por lo tanto la densidad aumenta en las pistas mas centrales

Discos Rígidos

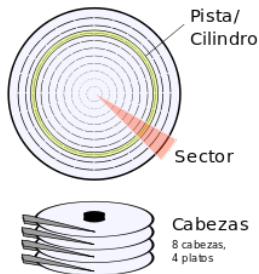
Estructura

Discos Rígidos

Estructura



- Ambas caras de un plato son útiles
- Un disco tiene varios platos apilados.
- Hay un cabezal por cada cara de cada plato.



Discos Rígidos

Acceso Directo

Discos Rígidos

Acceso Directo



- (1) Ubicar el cabezal sobre el cilindro buscado
- (2) Esperar a que el sector se ubique debajo del cabezal

Discos Rígidos

¿Como se direcciona un sector?

Discos Rígidos

¿Como se direcciona un sector?



Identificar el inicio de la pista y la delimitación de cada sector

Discos Rígidos

¿Como se direcciona un sector?



Identificar el inicio de la pista y la delimitación de cada sector



Incluir información de control



Ejemplo: Lectura de un sector del disco

Lectura de un sector del disco

Suponer un disco de 1000 pistas, 100 sectores, 4 caras con sectores de 512 bytes.

Leer el sector 25 de la pista 300 de la cara 3 y copiar el contenido del mismo en memoria a partir de la dirección 7000h.

El programa deberá dar al disco:

- 1 las coordenadas (P/C/S) del sector a leer
- 2 el comando que desea realizar: lectura.

ProgramaPrincipal:

EnviarComandos

EsperaDispositivo

TransferirDatos

Ejemplo: Lectura de un sector del disco

Puerto	Sentido de la com	Significado
4000h	CPU → Disco	Nro Pista
4001h	CPU → Disco	Nro Cara
4002h	CPU → Disco	Nro Sector
4003h	CPU → Disco	Comando: 1 Leer 2 Escribir
4003h	Disco → CPU	Estado: 4 No listo 5 Listo para transferir
4004h	Disco → CPU	Datos leídos
4004h	CPU → Disco	Datos a escribir

Ejemplo: Lectura de un sector del disco

EnviarComandos:

Ejemplo: Lectura de un sector del disco

EnviarComandos:

OUT 0x4000,0x012C; pista 300

OUT 0x4001,0x0003; cara 3

OUT 0x4002,0x0019; sector 25

OUT 0x4003,0x0001; 1=Lectura

EsperaDispositivo:

Ejemplo: Lectura de un sector del disco

EnviarComandos:

```
OUT 0x4000,0x012C; pista 300  
OUT 0x4001,0x0003; cara 3  
OUT 0x4002,0x0019; sector 25  
OUT 0x4003,0x0001; 1=Lectura
```

EsperaDispositivo:

```
IN R0, 0x4003  
CMP R0, 0x0004  
JE EsperaDispositivo
```

TransferirDatos:

Ejemplo: Lectura de un sector del disco

EnviarComandos:

```
OUT 0x4000,0x012C; pista 300  
OUT 0x4001,0x0003; cara 3  
OUT 0x4002,0x0019; sector 25  
OUT 0x4003,0x0001; 1=Lectura
```

EsperaDispositivo:

```
IN R0, 0x4003  
CMP R0, 0x0004  
JE EsperaDispositivo
```

TransferirDatos:

```
MOV R1, 0x7000  
MOV R2, 0x0100; 256 celdas de 16 bits  
= 512bytes  
arriba: CMP R2, 0x0000 JE fin IN R0,  
0x4004  
MOV [R1], R0  
ADD R1, 0x0001  
SUB R2, 0x0001  
JNE arriba  
fin: RET
```

Conclusiones

¿Preguntas?