Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores

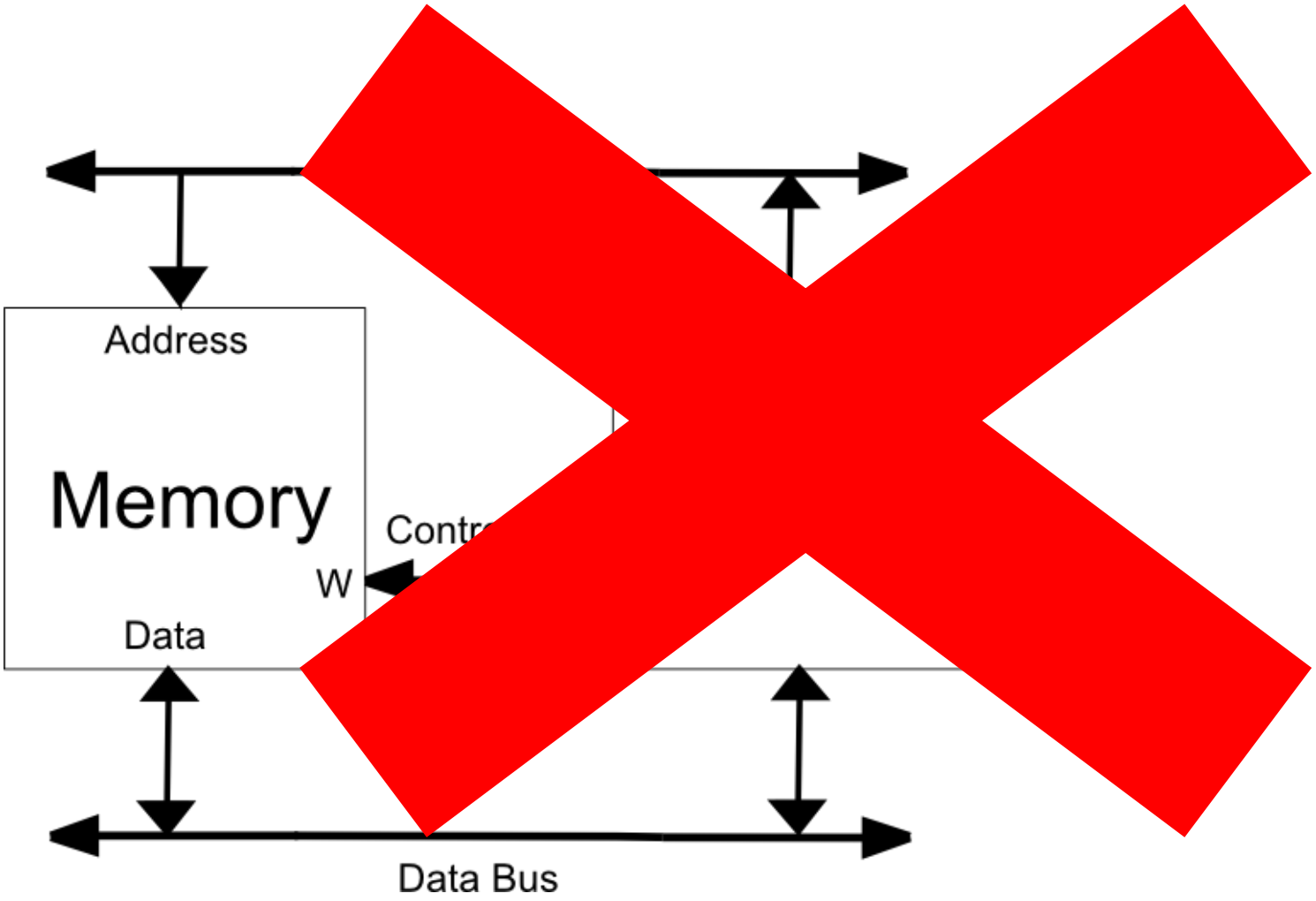
Conexión de CPU y Memoria con I/O

Profesor: Hans-Albert Löbel

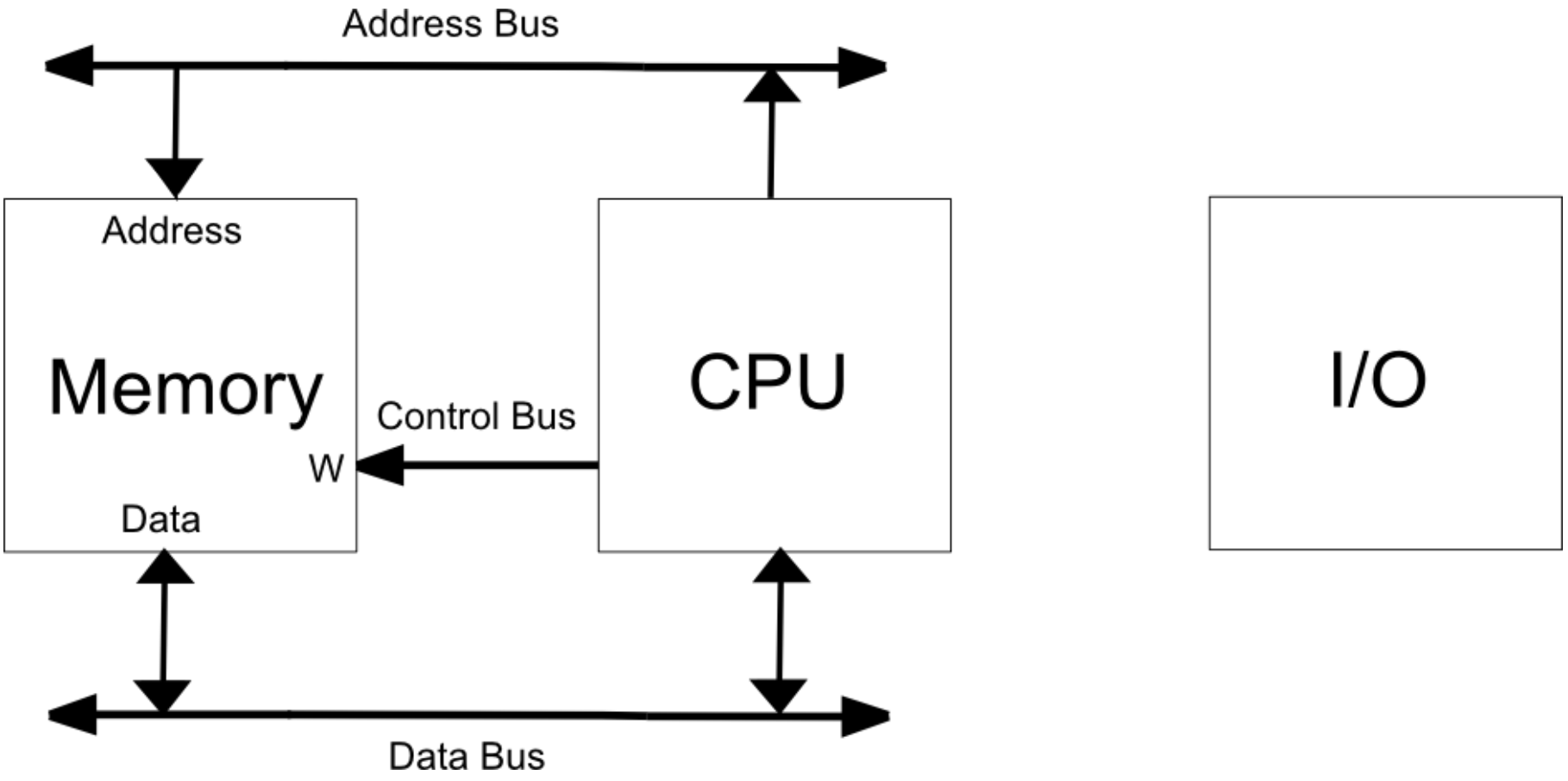
¿Qué nos falta?

- Ya aprendimos como construir un computador básico y cómo programarlo.
- Pero este computador aun dista de tener todas las funcionalidades de un computador estándar.
- Nos falta estudiar la comunicación con usuarios, entre las partes, con otros dispositivos y con otros computadores

Todo computador está compuesto por 2 elementos principales: CPU y memoria



Todo computador está compuesto por 3 elementos principales: CPU, memoria y dispositivos de I/O

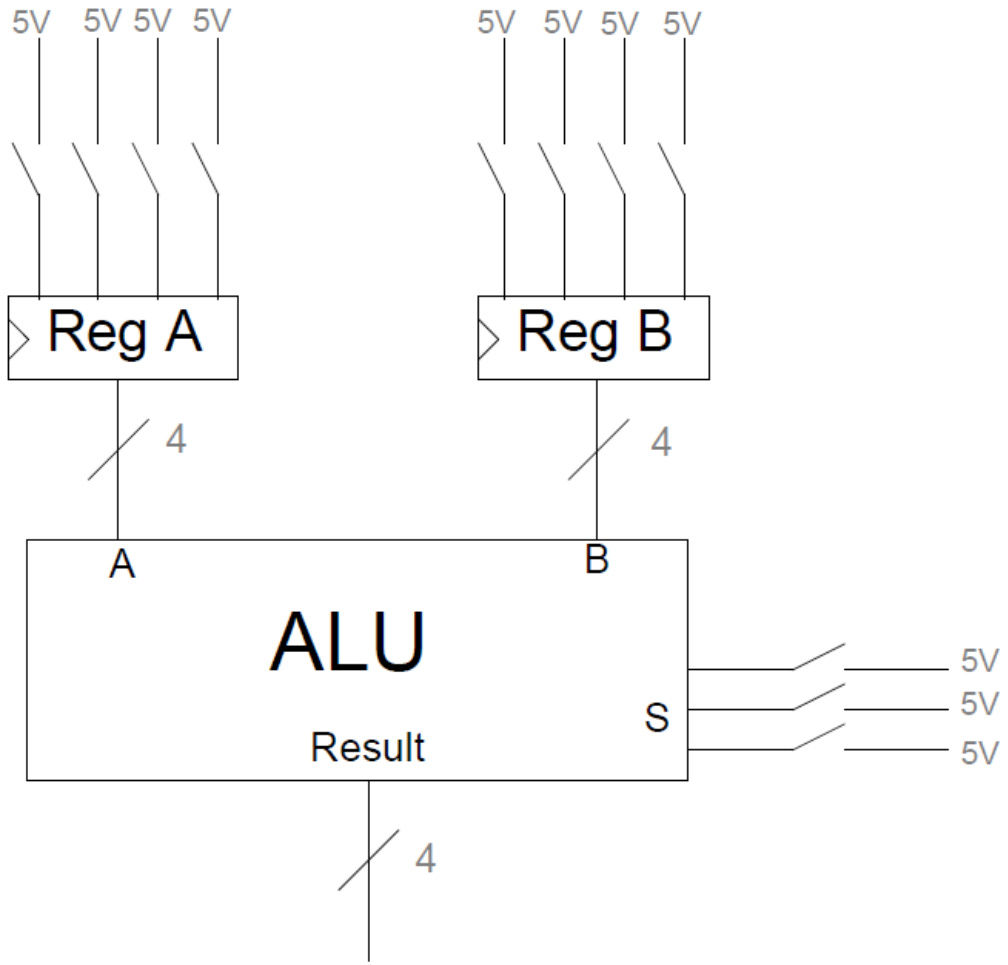


Comunicación depende de los "sentidos" del receptor

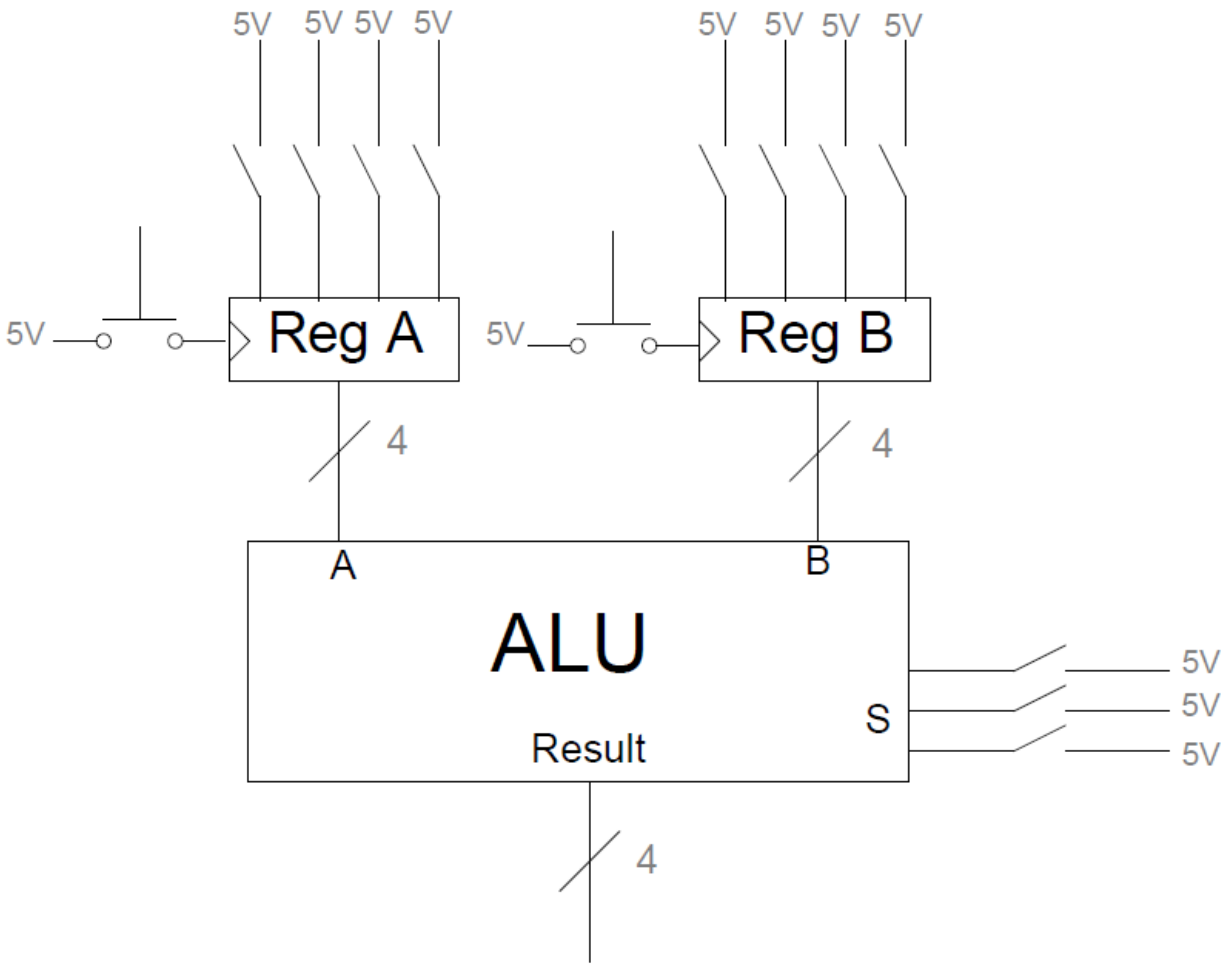
	Computador -> Humano	Humano -> Computador
Visión	Display	Cámara
Audición	Parlantes	Micrófono
Tacto	Vibración	Interruptor, Teclado, Mouse

Dispositivos simples de entrada

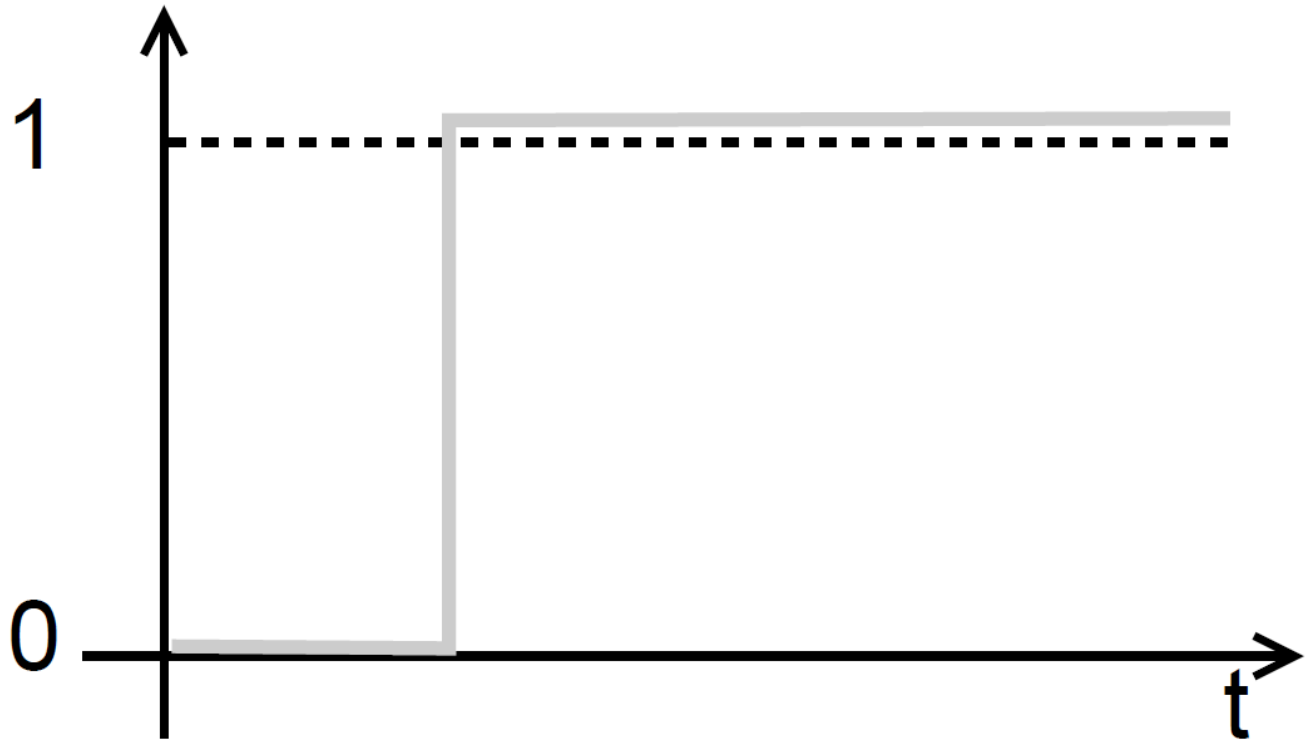
Interruptores para ingresar datos en registros y operación en ALU



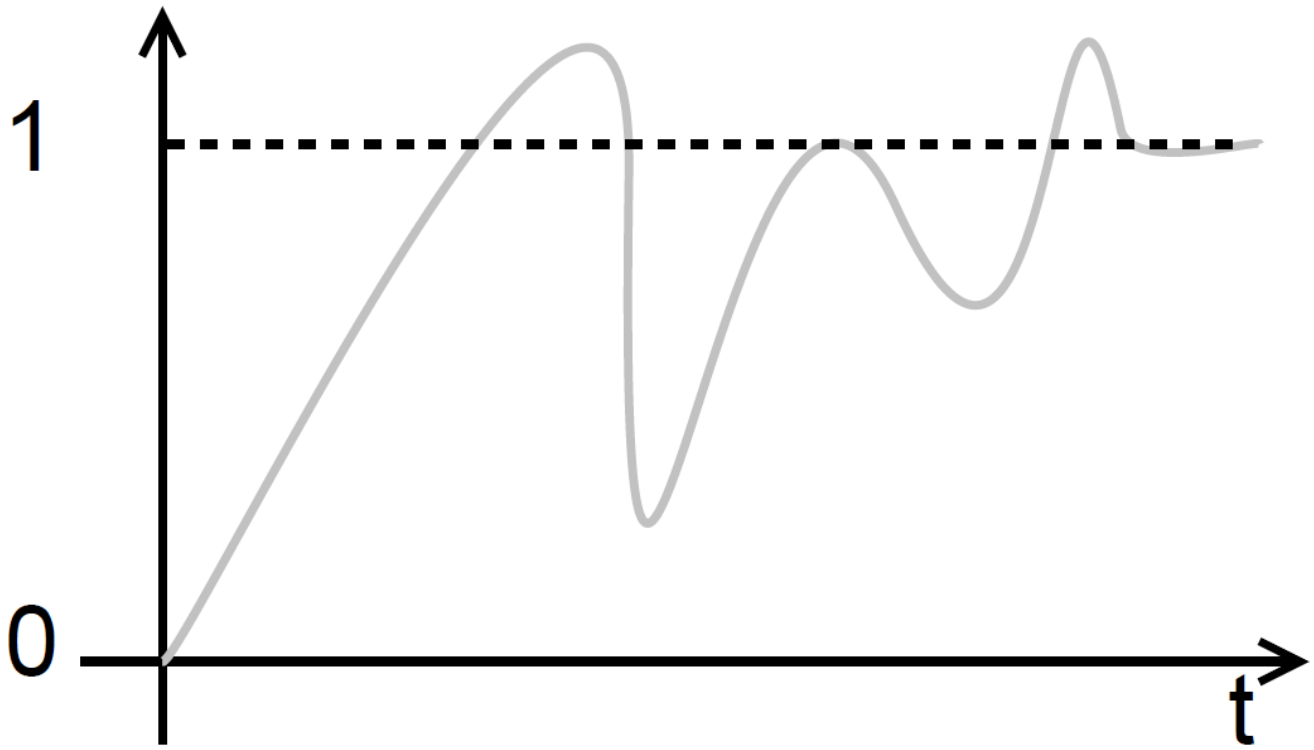
Botones para controlar los registros



Interruptor ideal presenta el comportamiento deseado

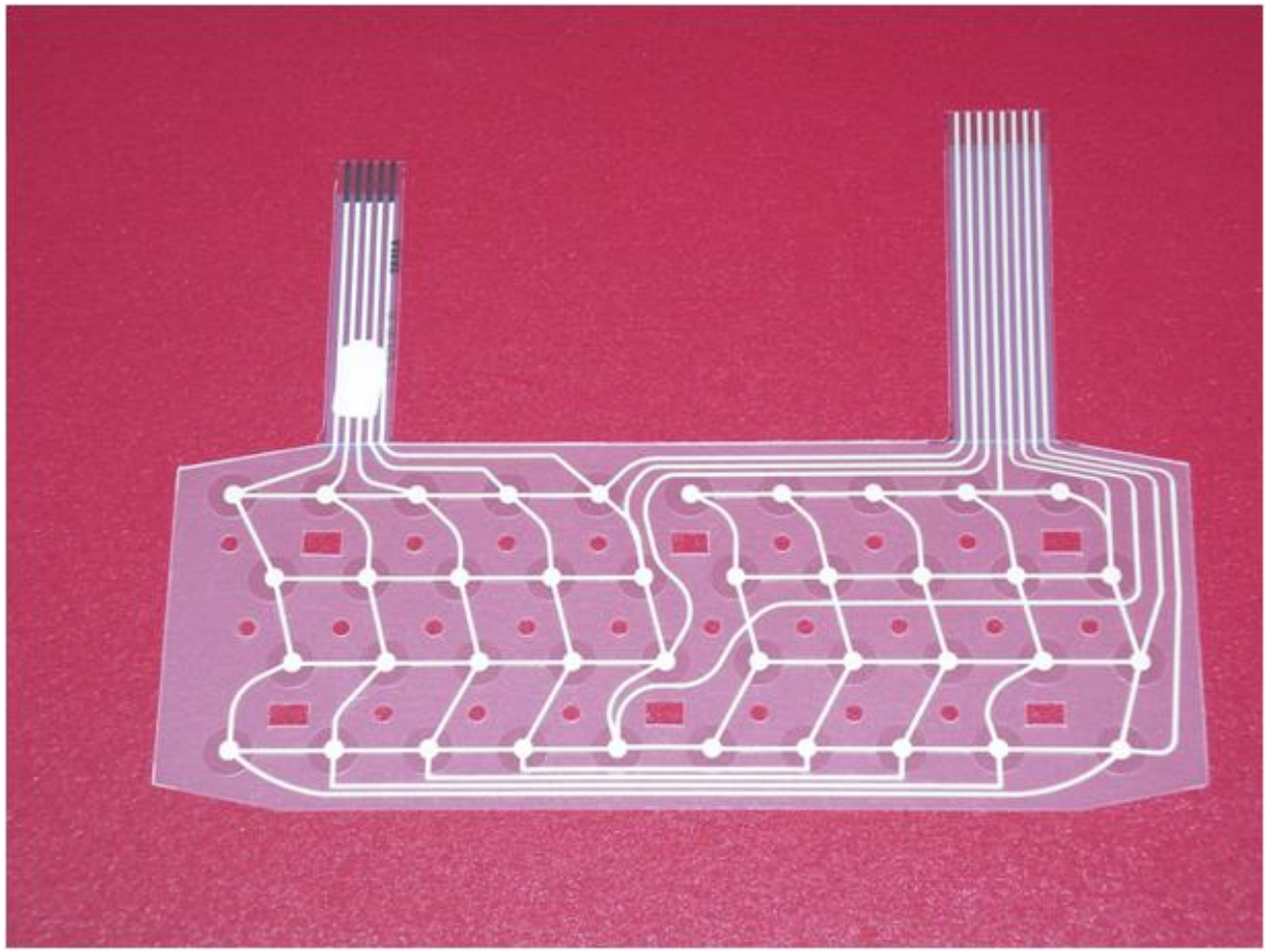


Interruptor real oscila alrededor del valor deseado (transiente)



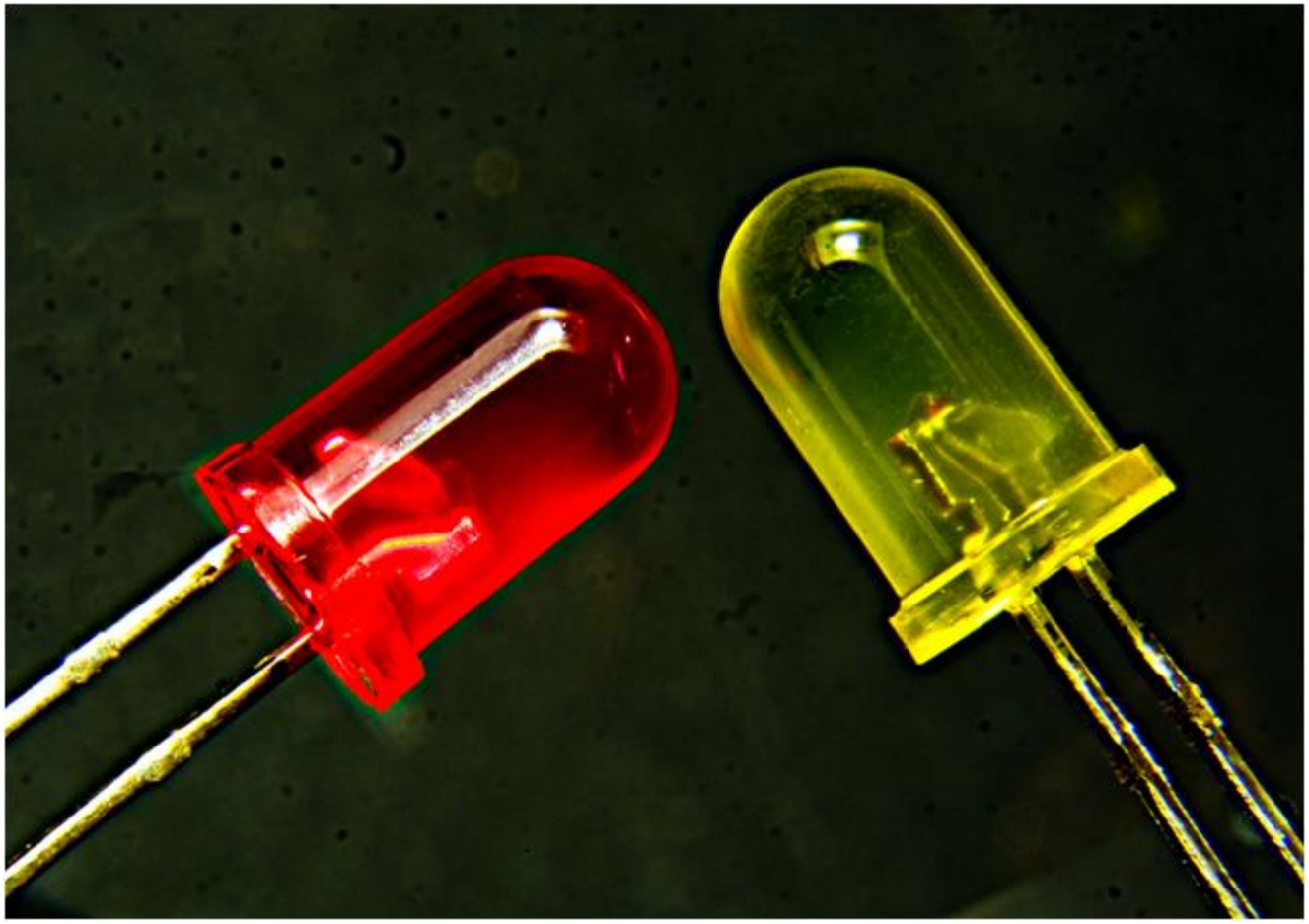
Un teclado es un conjunto de botones

Cada botón se asocia a una fila y una columna

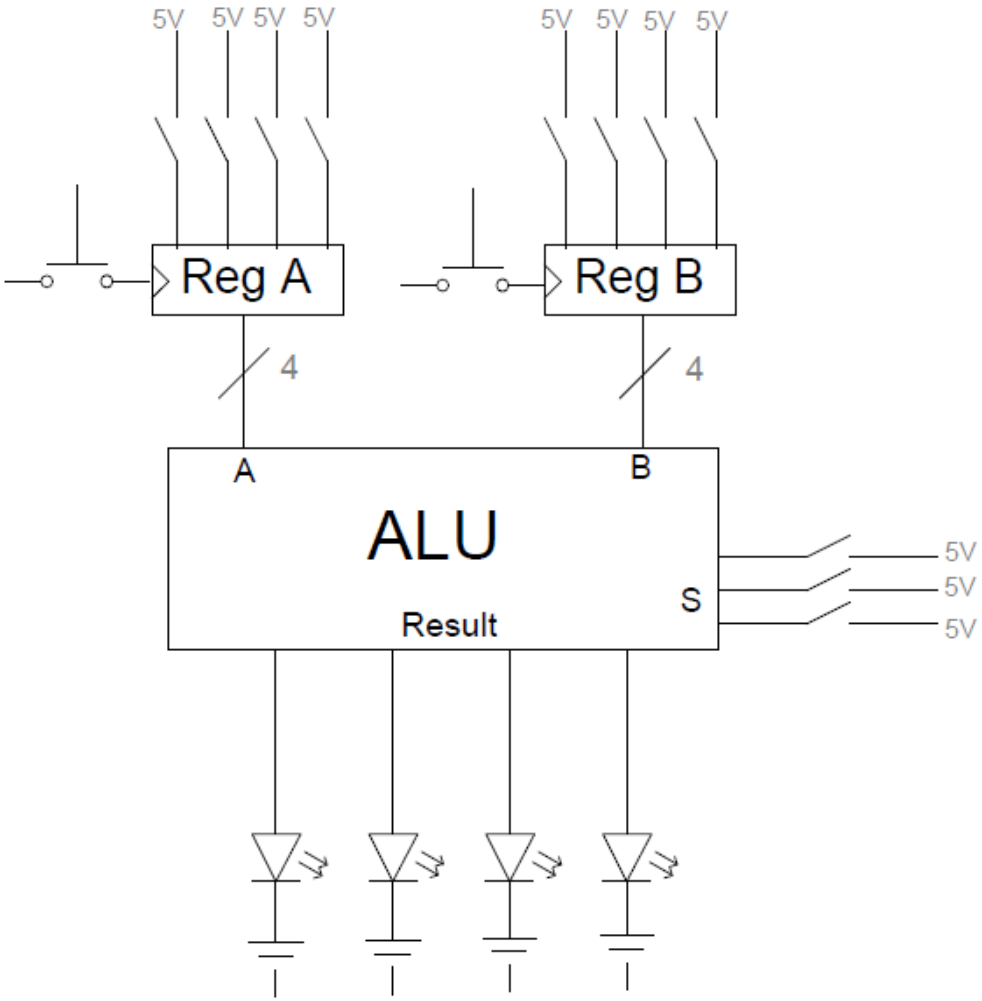


Dispositivos simples de salida

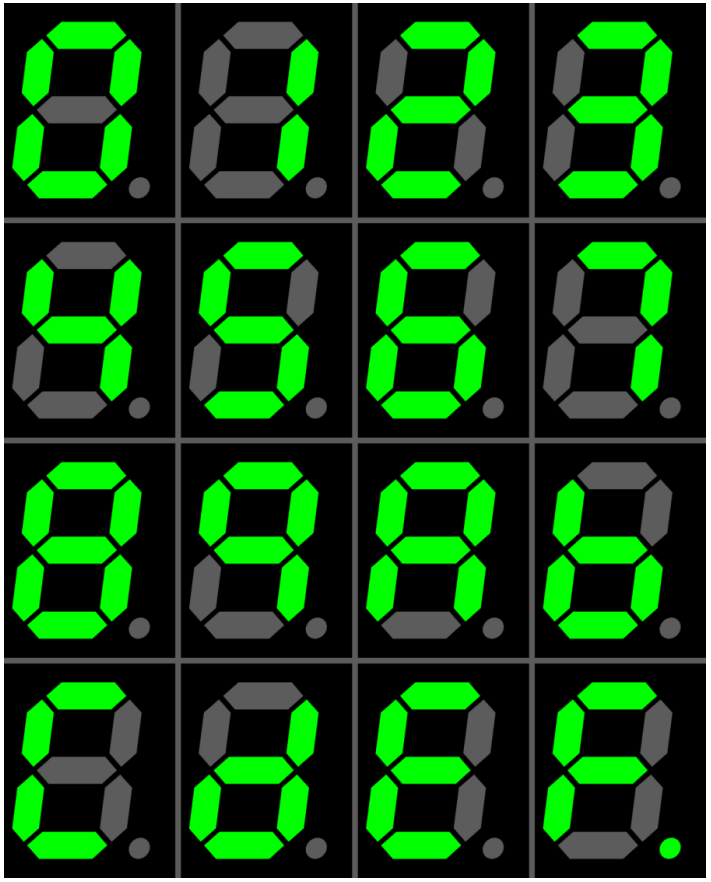
Los LED son los dispositivos de salida más básicos



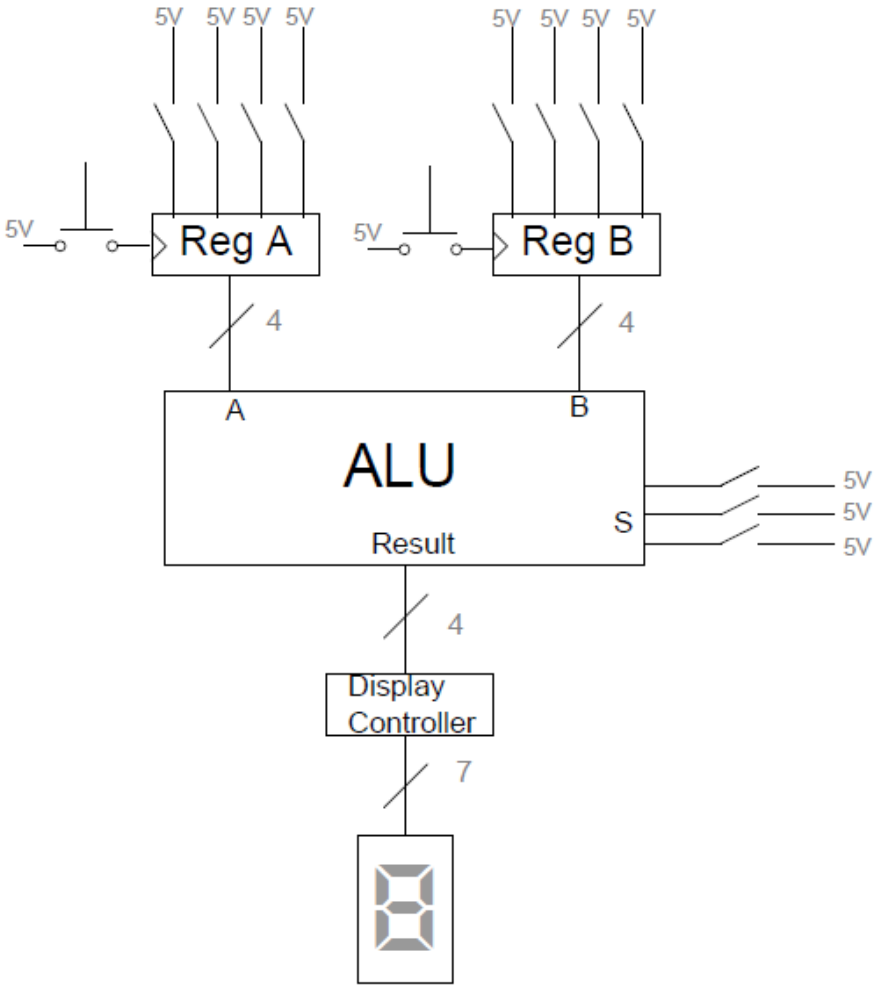
LEDs pueden conectarse directamente a la ALU



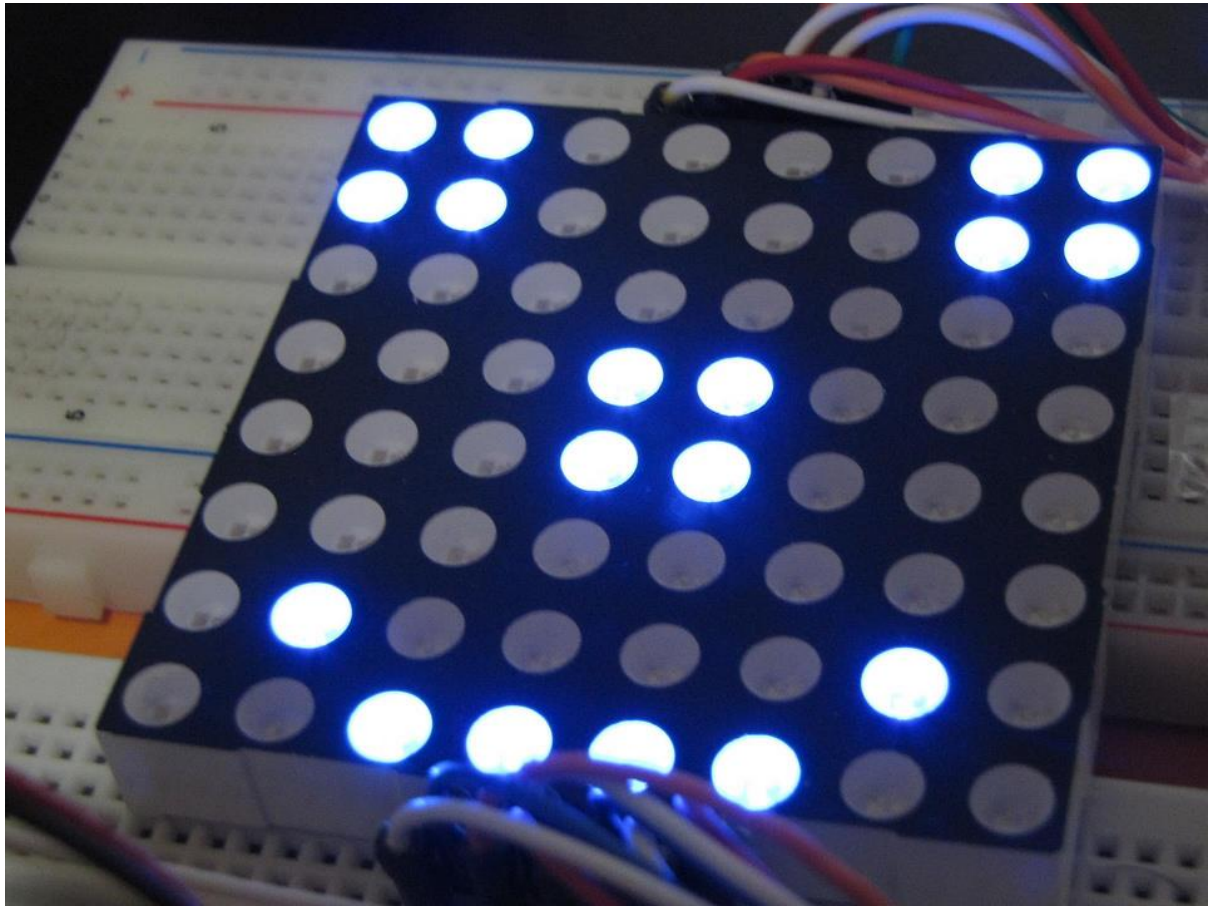
El display de 7 segmentos permite entregar feedback más “humano”



El display se conecta mediante un *Display Controller* a la ALU

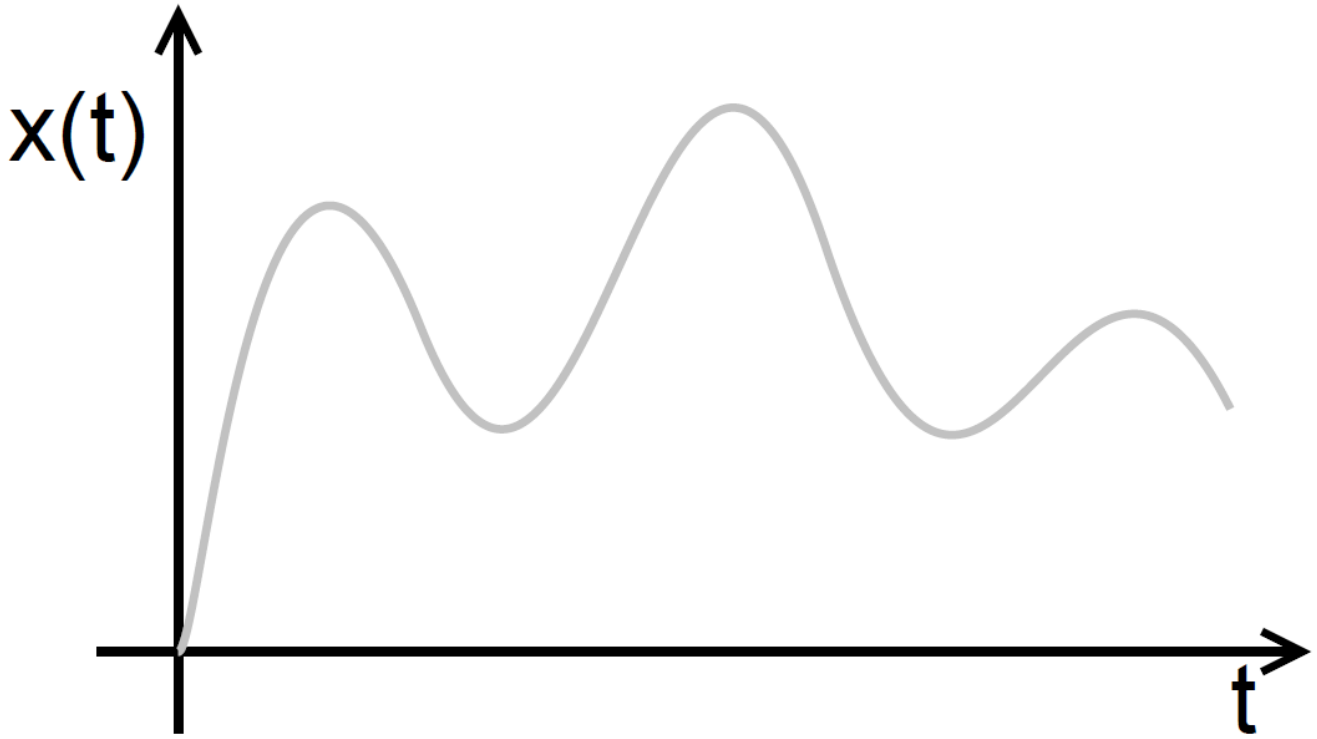


Matrices de LED son el paso previo a dispositivos más complejos (pantallas)

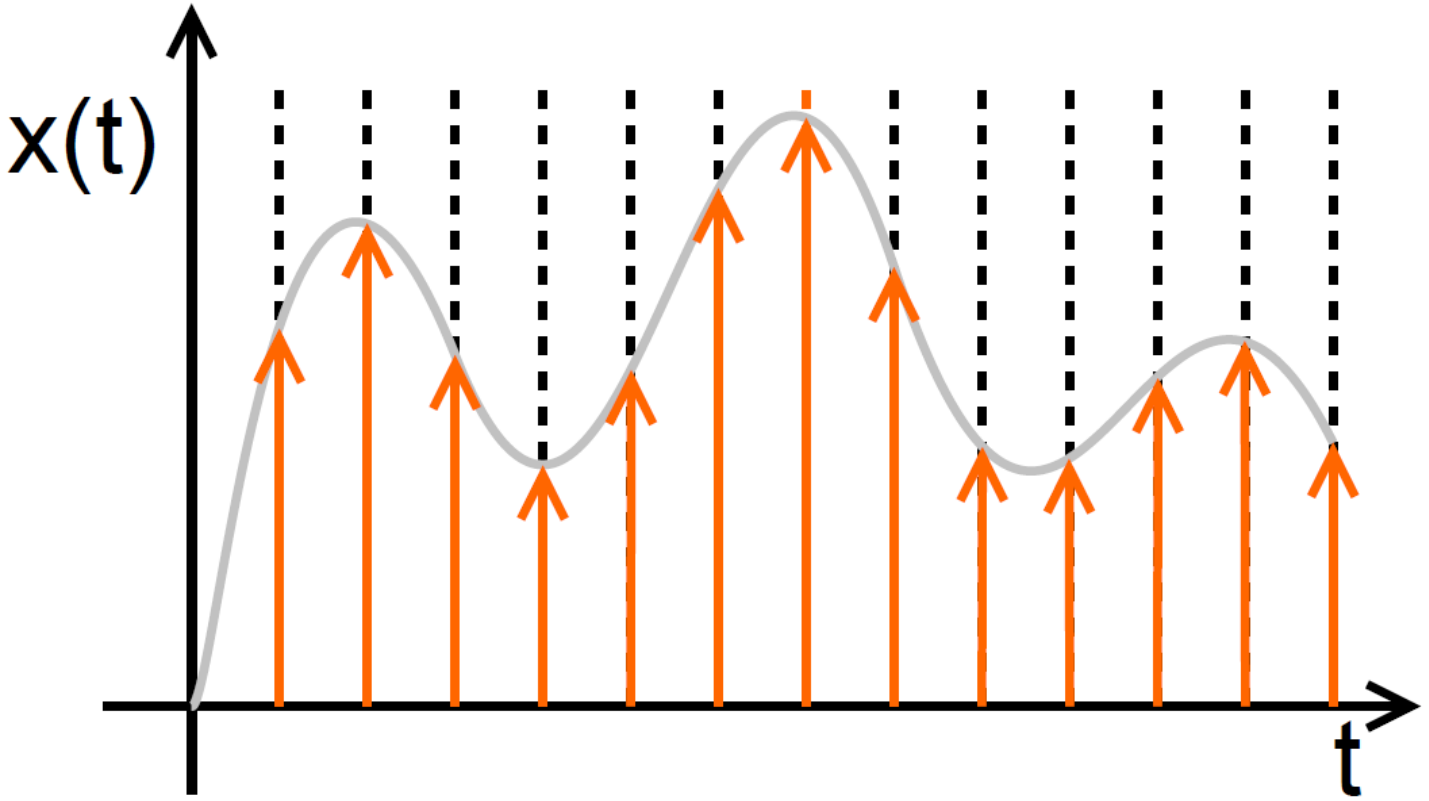


Interacción avanzada: señales continuas

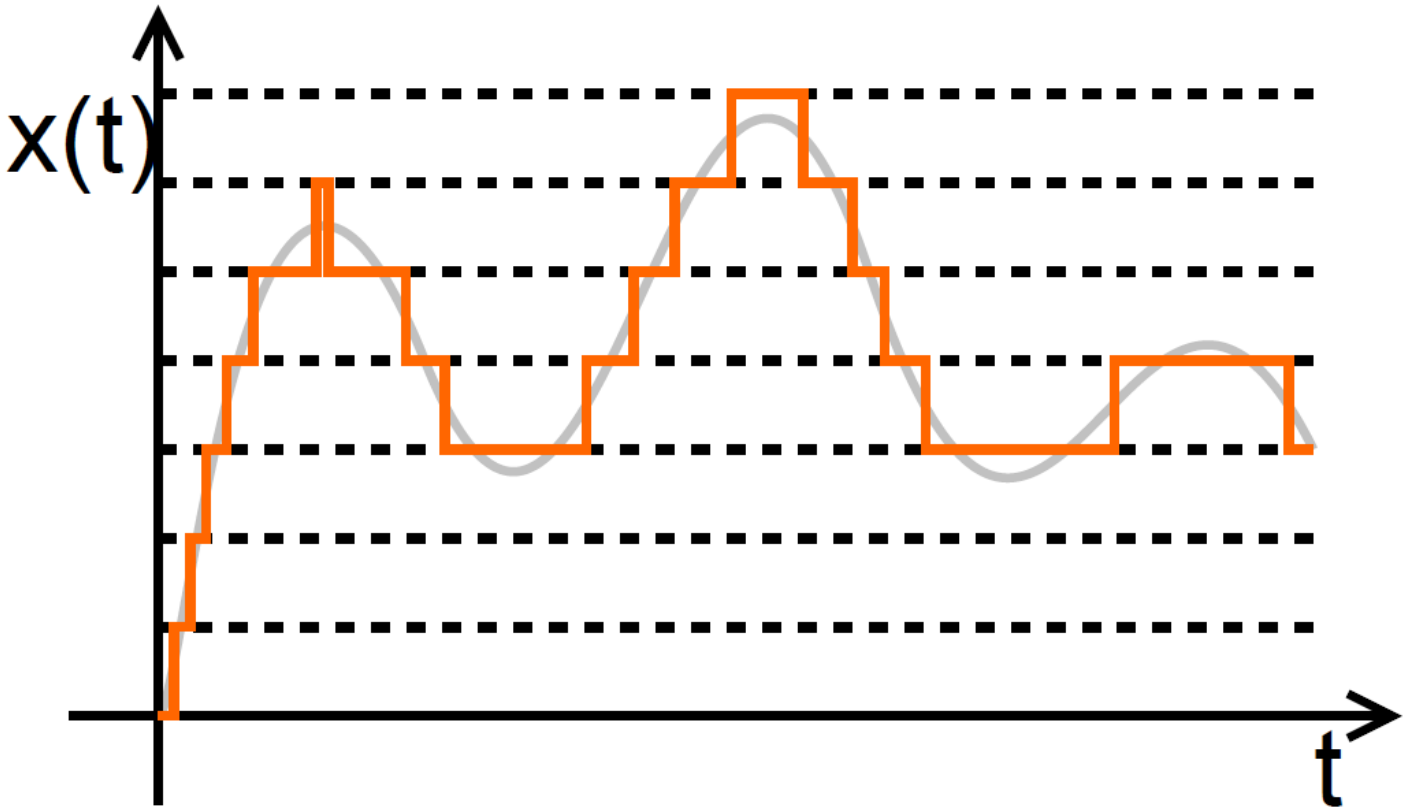
Señales continuas no pueden ser almacenadas por un computador



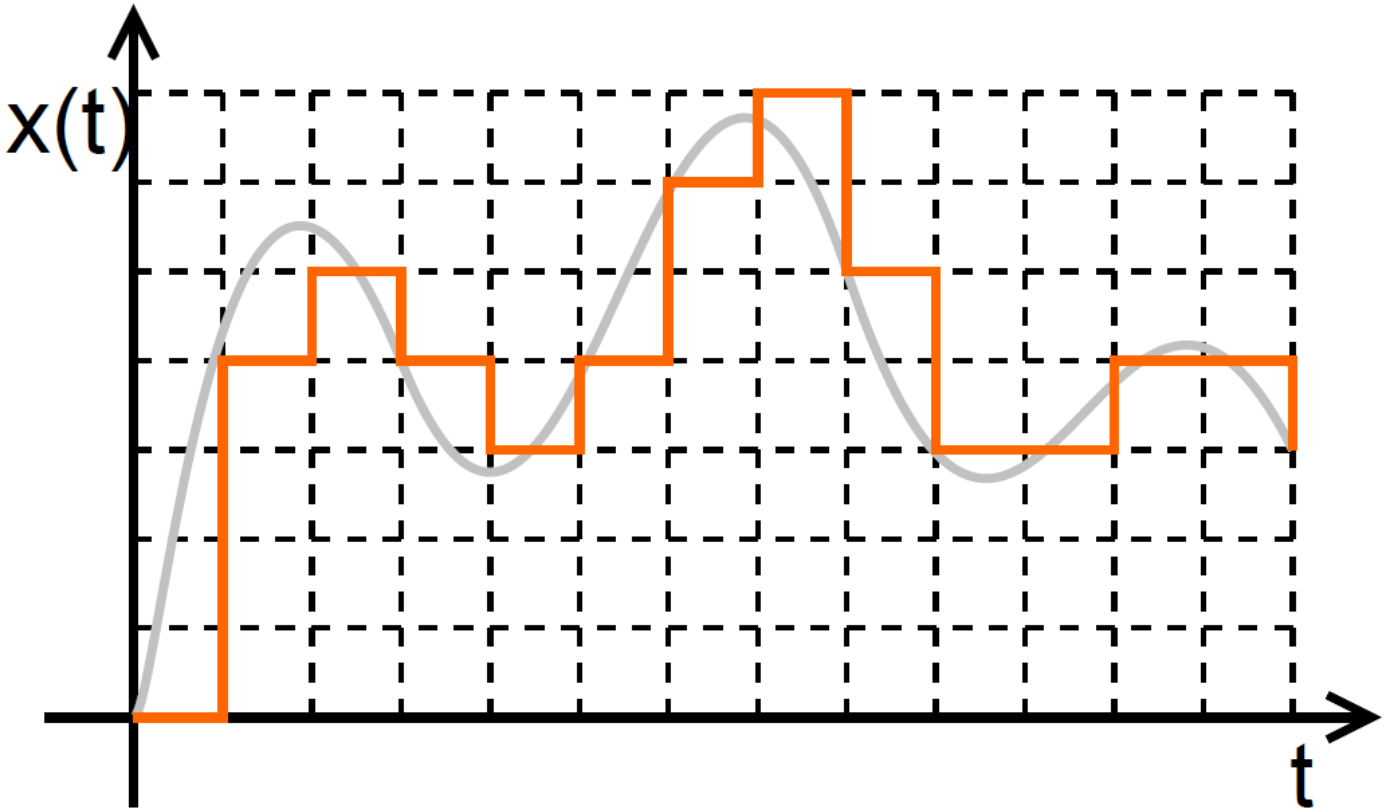
Muestreo permite manejar la continuidad temporal de la señal



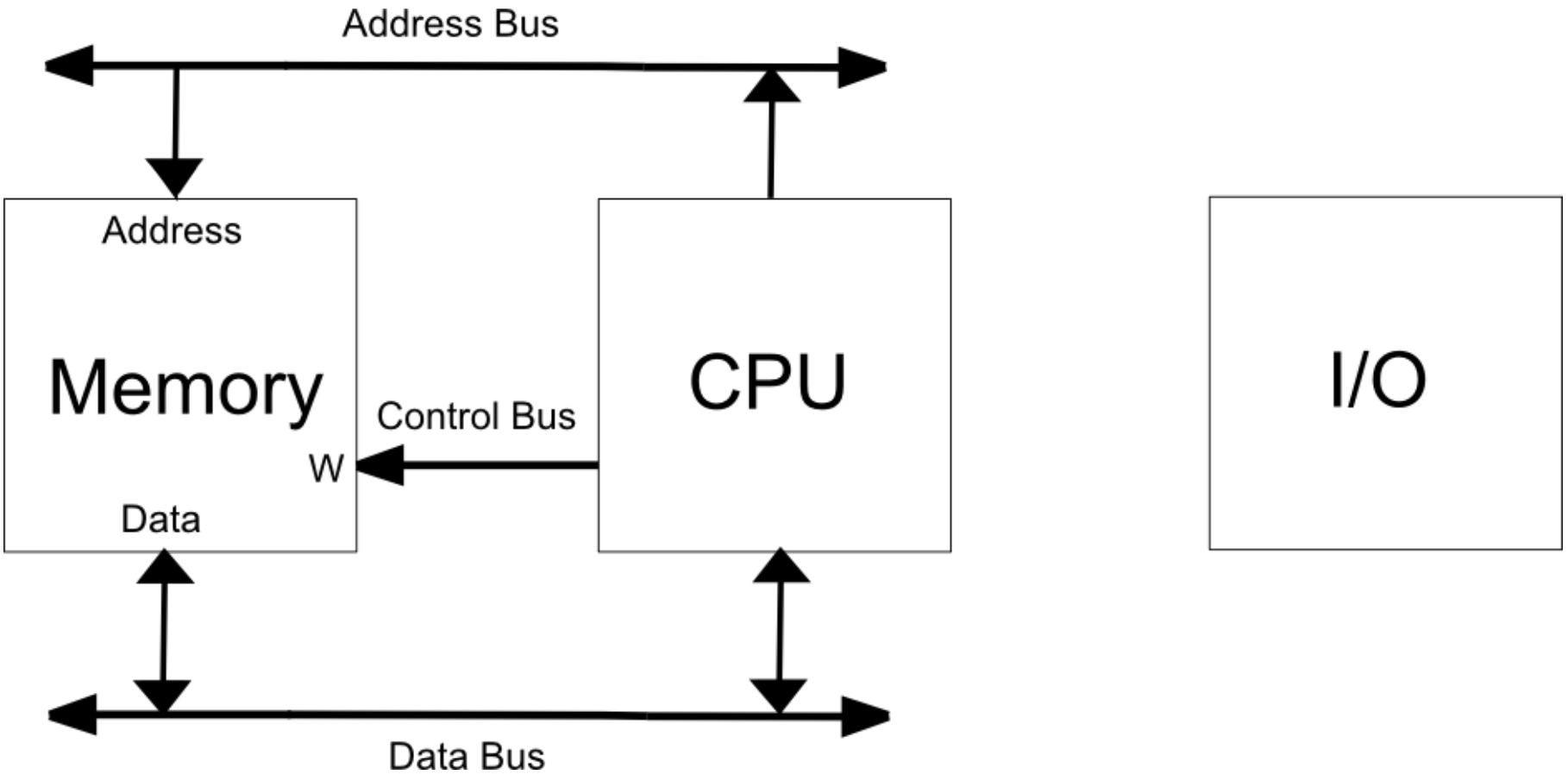
Cuantización permite manejar la continuidad de la amplitud de la señal



La señal digital (discreta) se obtiene al muestrear y cuantizar la señal original



Todo computador está compuesto por 3 elementos principales: CPU, memoria y dispositivos de I/O



Dispositivos comparten estructura común

- A pesar de existir una gran variedad, la mayoría de los dispositivos tienen 2 componentes bien definidos:

    1. Elementos electromecánicos: realizan las operaciones de interacción.
    2. Controladores: regulan el comportamiento de los componentes electromecánicos y también la comunicación con el resto del computador.

¿Cuándo se invento el mouse?

a) 1950s

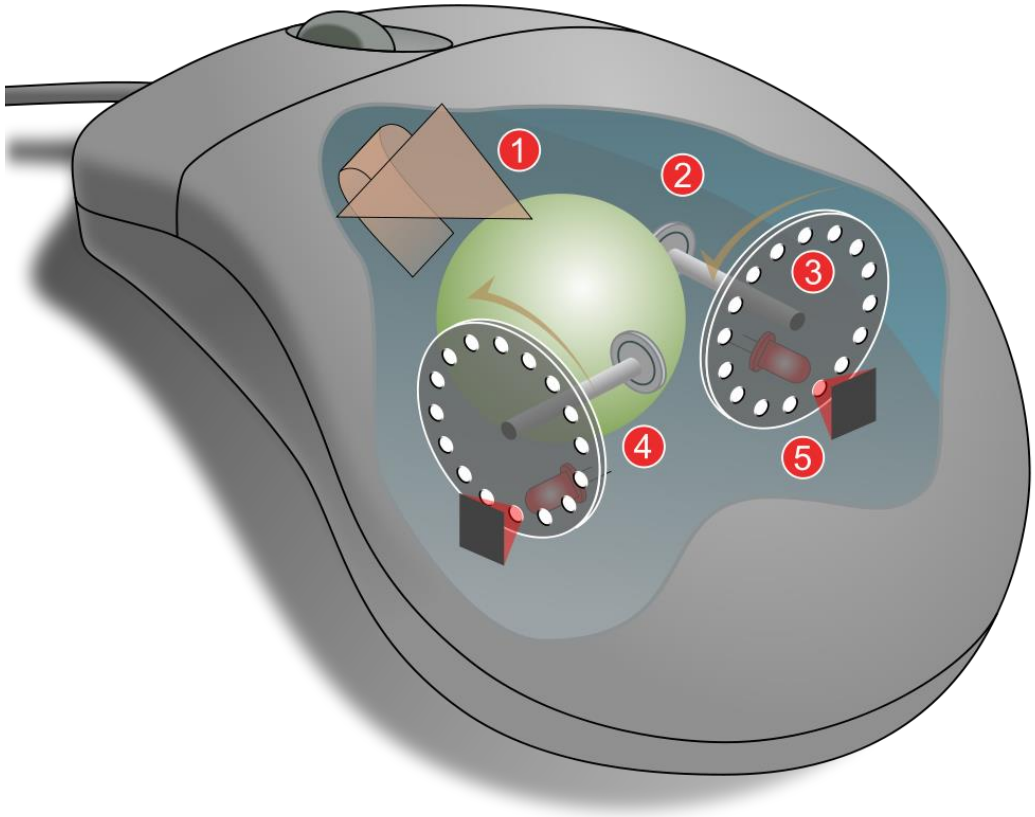
b) 1960s

c) 1970s

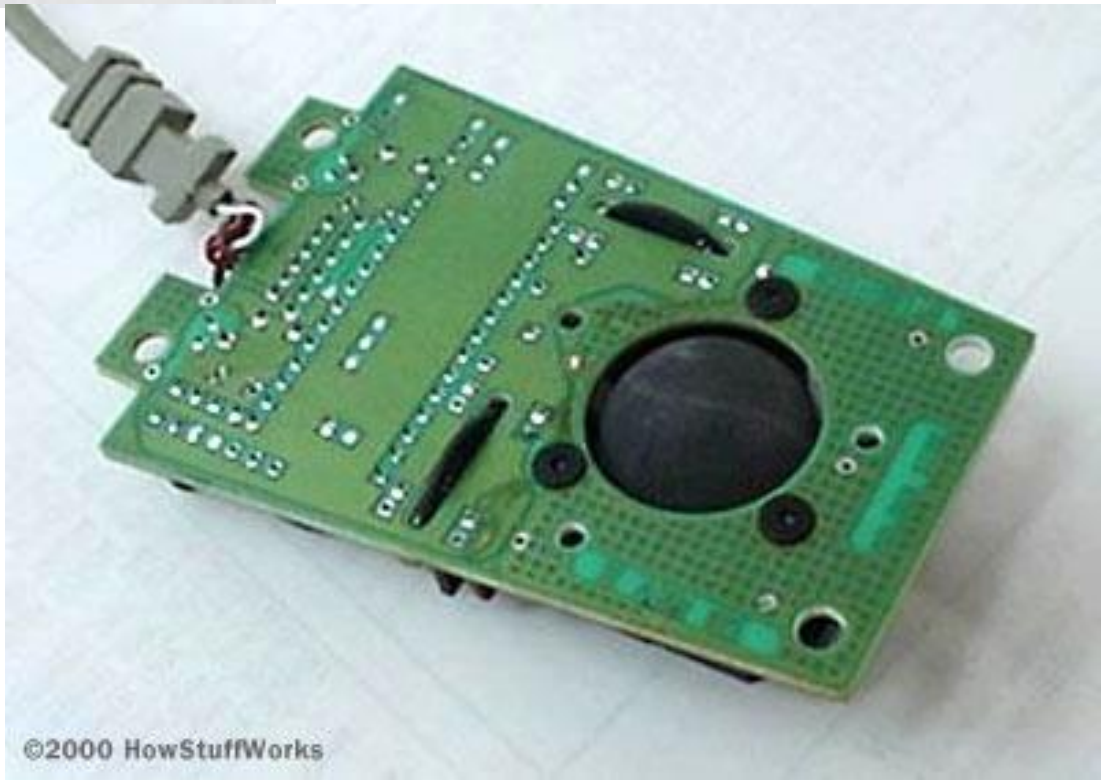
d) 1980s

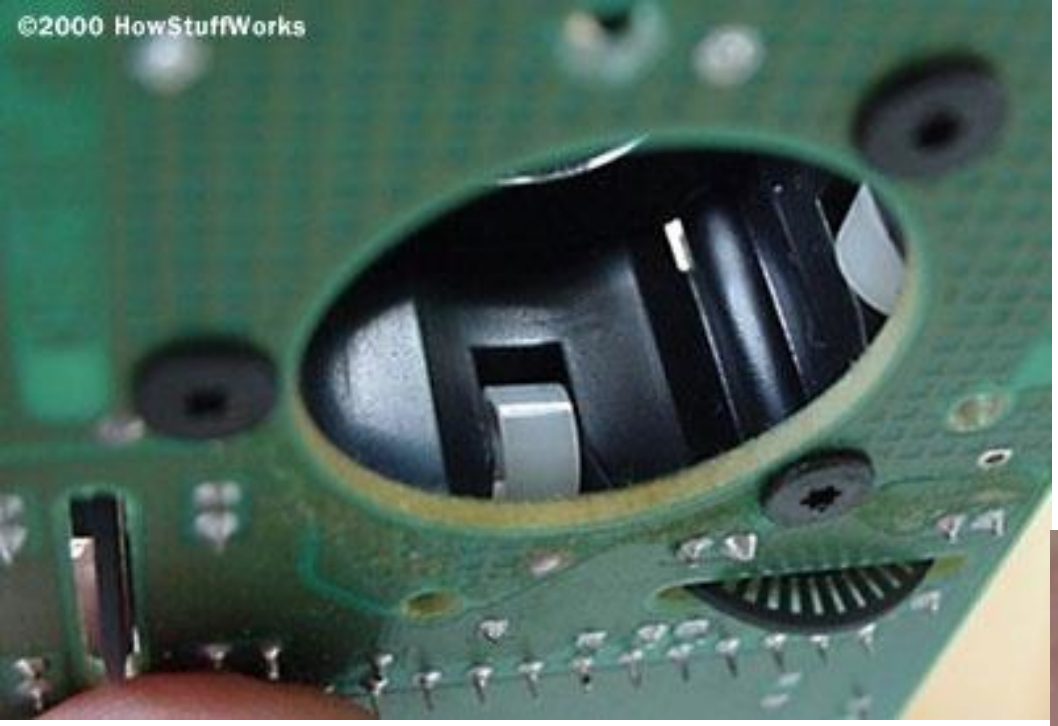
e) 1990s

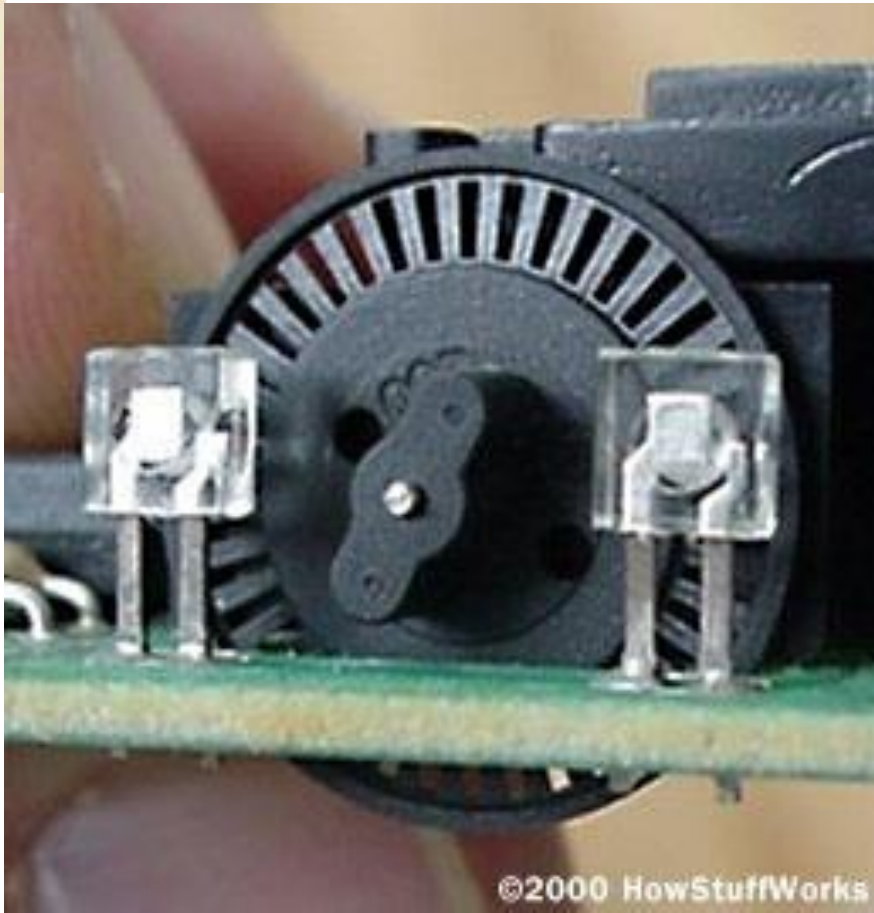
Mouse discretiza la señal continua del movimiento

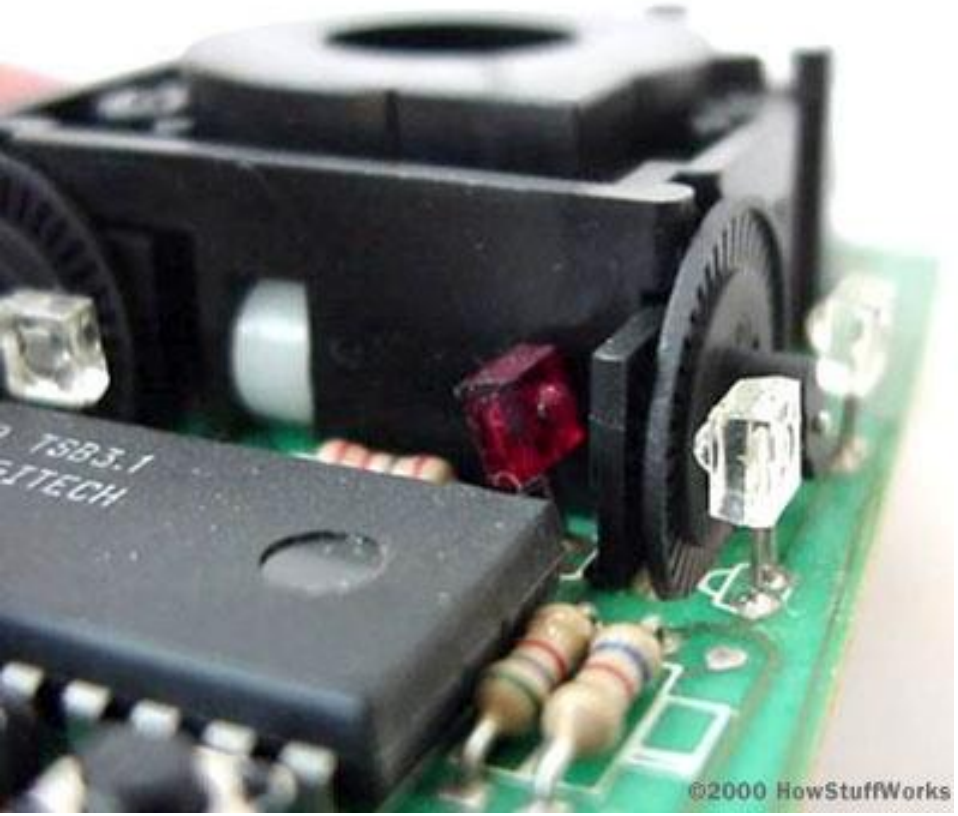



©2000 HowStuffWorks


©2000 HowStuffWorks


©2000 HowStuffWorks

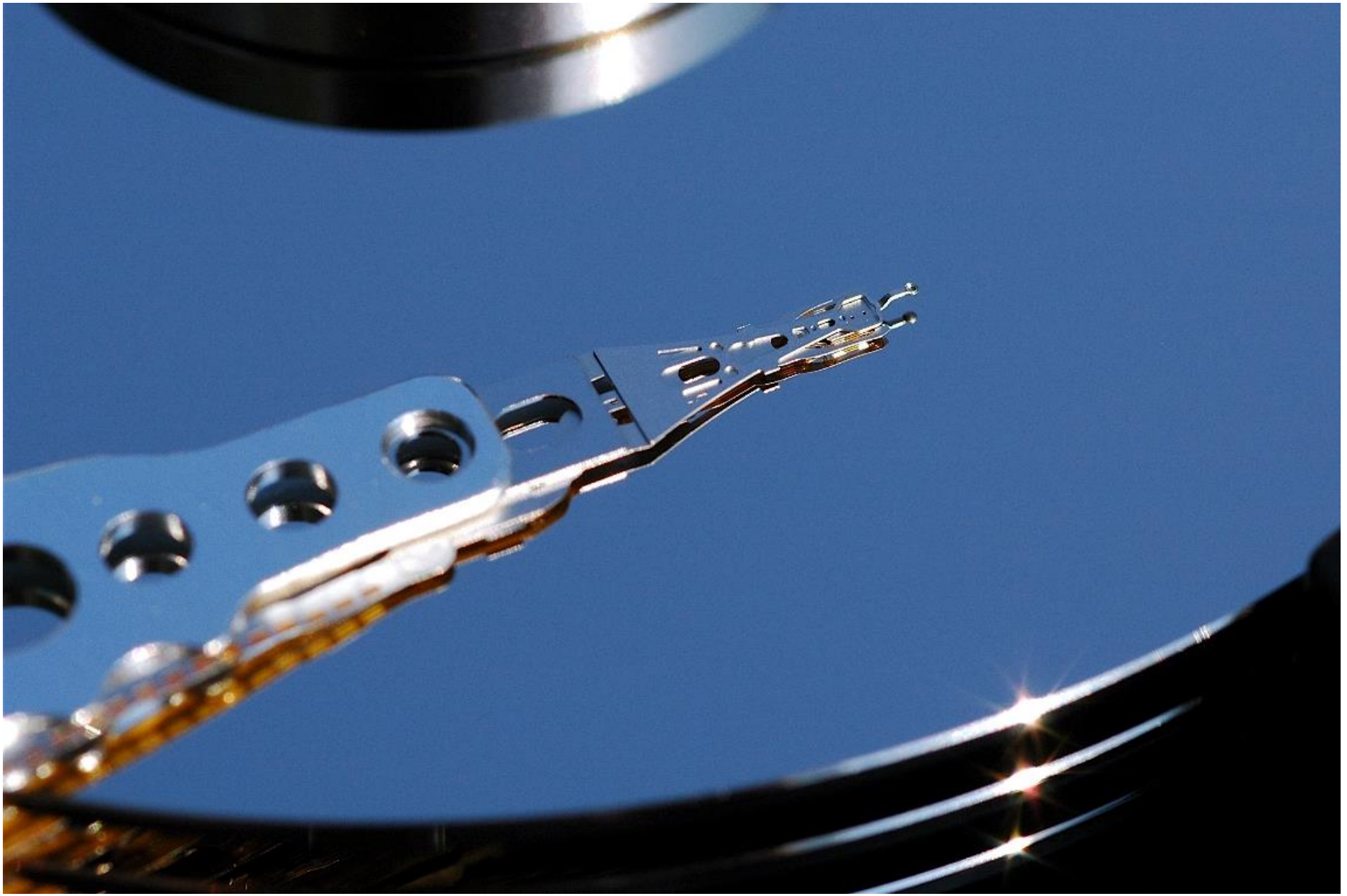

©2000 HowStuffWorks

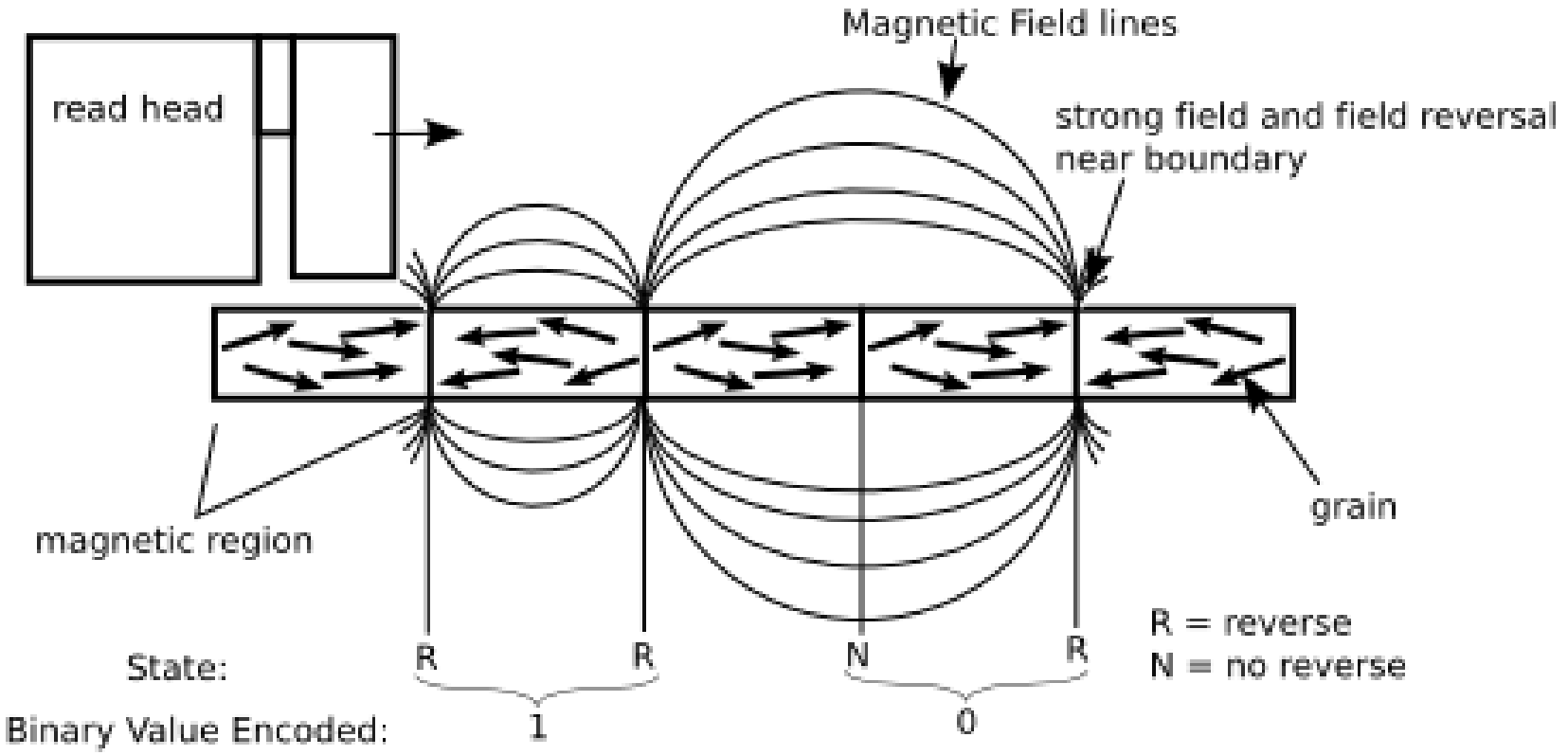

TSB3.1
©2000 HowStuffWorks

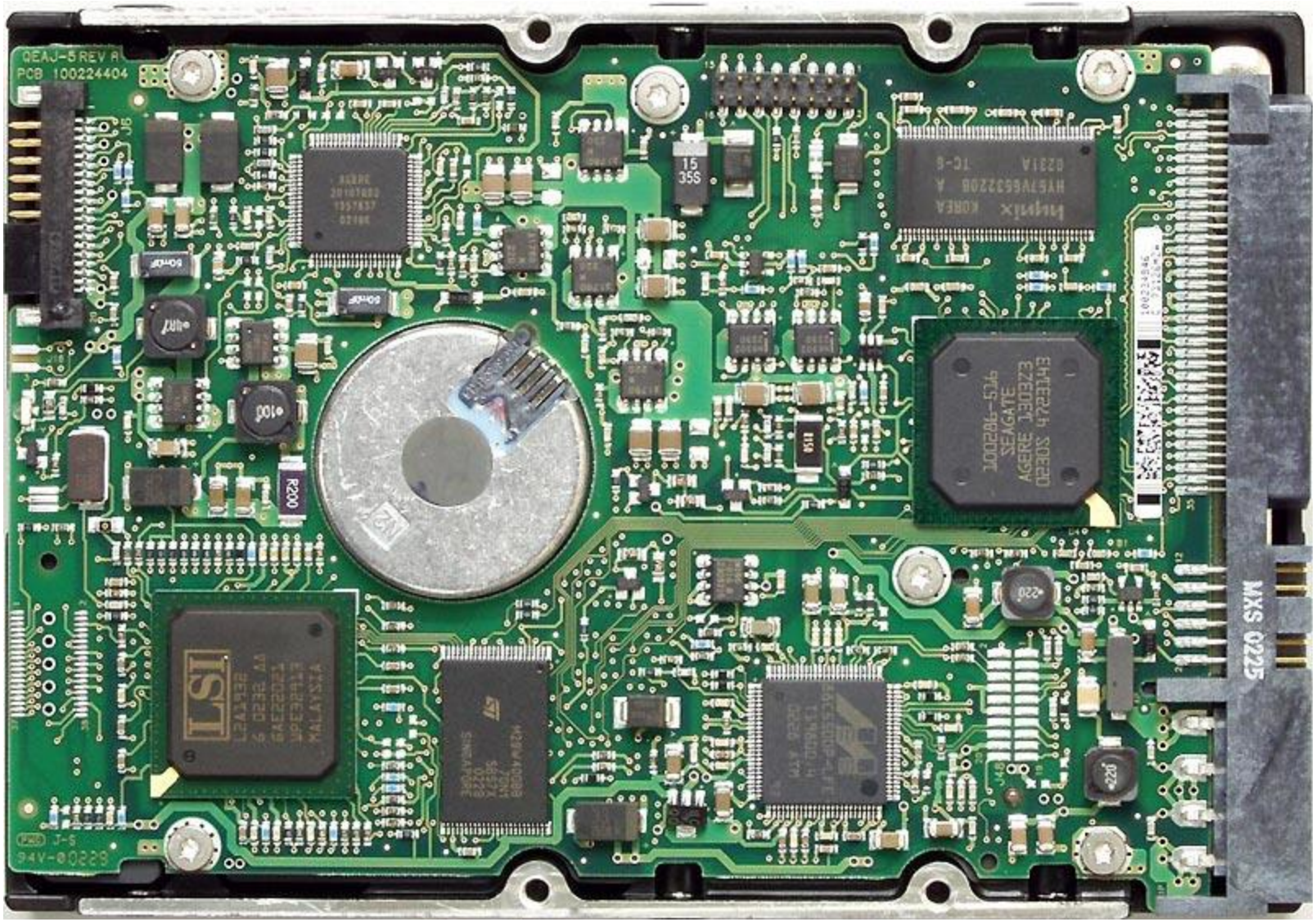

1989 TSB3.1
© LOGITECH
JG2
QLNN9218
©2000 HowStuffWorks

1956: Un disco duro de 5MB es cargado en un avión.

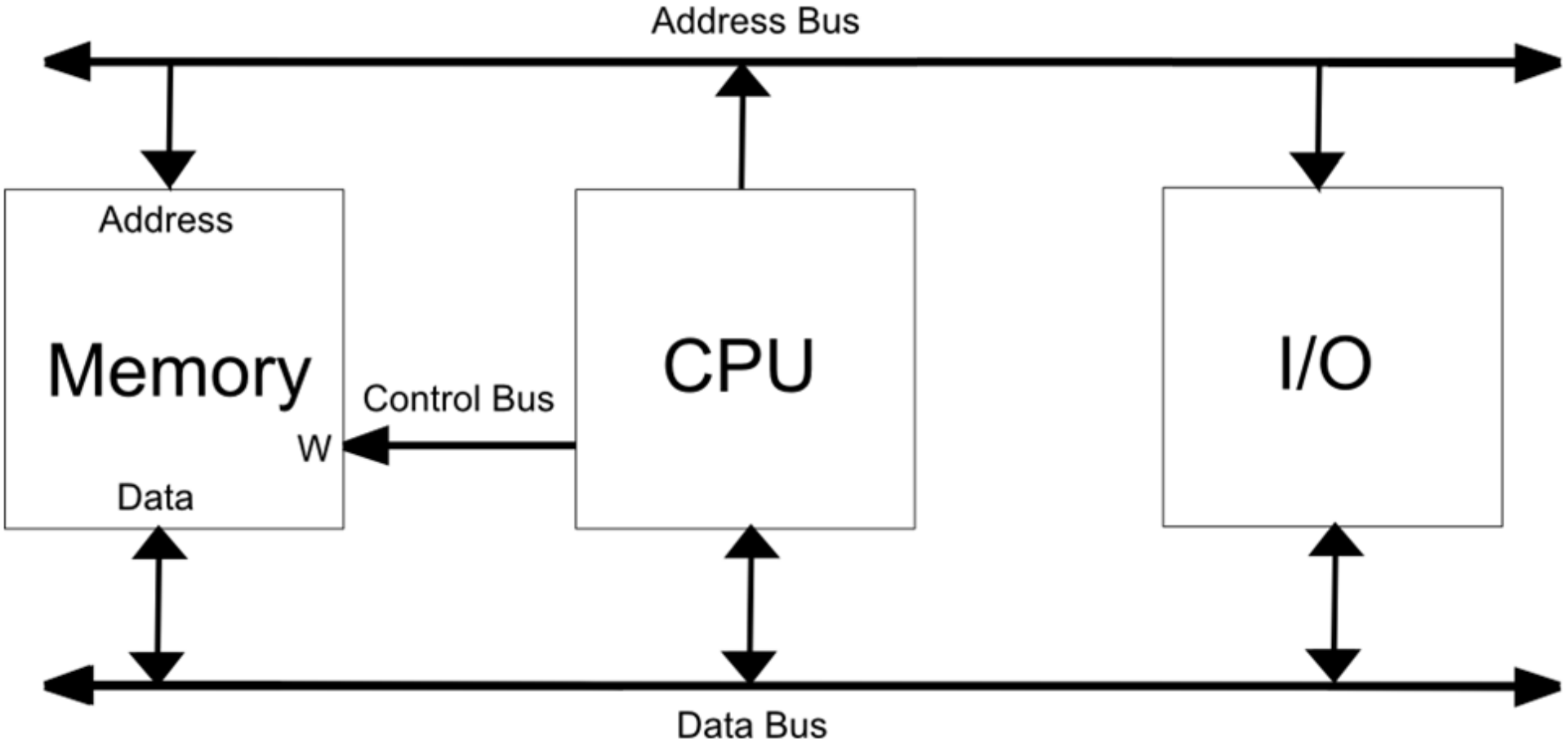

WARRANTY
VOID
IF REMOVED



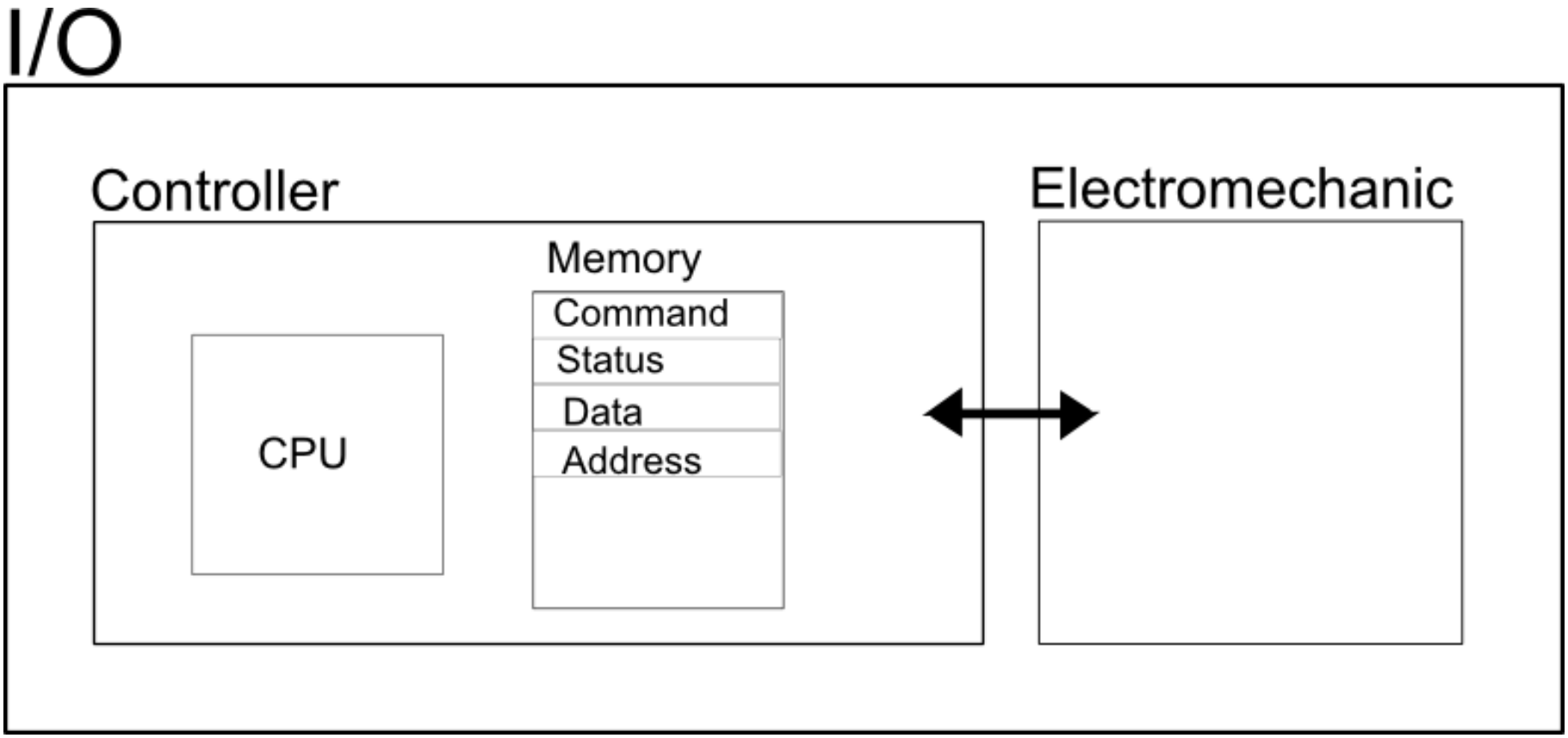

read head
Magnetic Field lines
strong field and field reversal
near boundary
magnetic region
grain
R = reverse
N = no reverse
State:
R
R
N
R
Binary Value Encoded:
1
0


QEAJ-5 REV A
PCB 100224404
LSI
MALAYSIA
MXS 0225

Todos los dispositivos que no sean CPU o memoria, y se comuniquen con ellos, son llamados dispositivos de entrada/salida (E/S o I/O)



Un dispositivo de I/O contiene un controlador que se encarga, entre otras cosas, de comunicarse con la CPU y la memoria y de controlar la parte electromecánica

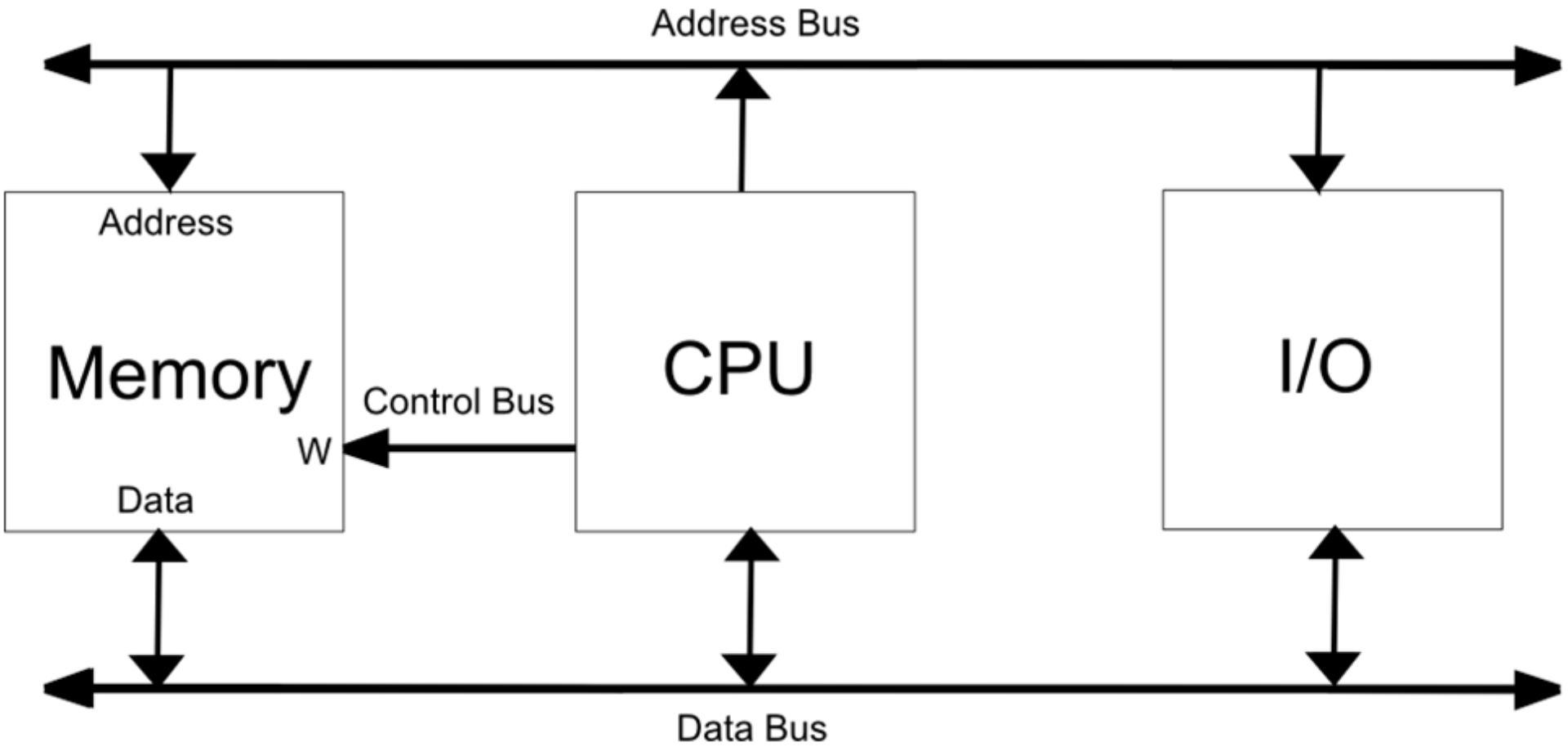


Dispositivos de I/O se comunican de manera distinta al resto de los elementos de un computador

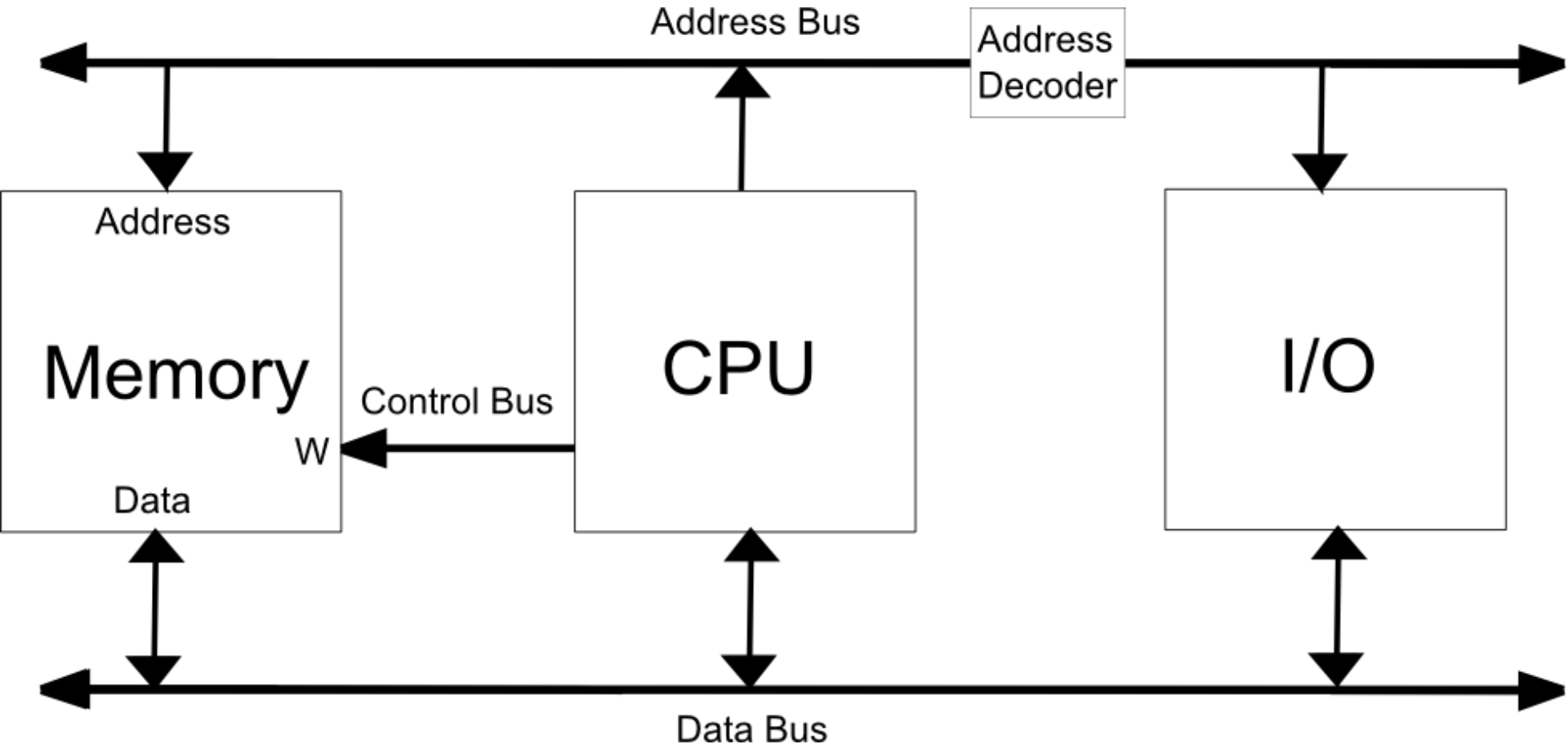
Al no existir señales de control explícitas para los dispositivos de I/O, debemos definir qué tipo de comunicación se llevará a cabo entre CPU, memoria y estos:

1. Comunicación de comandos: CPU -> I/O
2. Comunicación de estado: I/O -> CPU
3. Transferencia de datos: Memoria <-> I/O

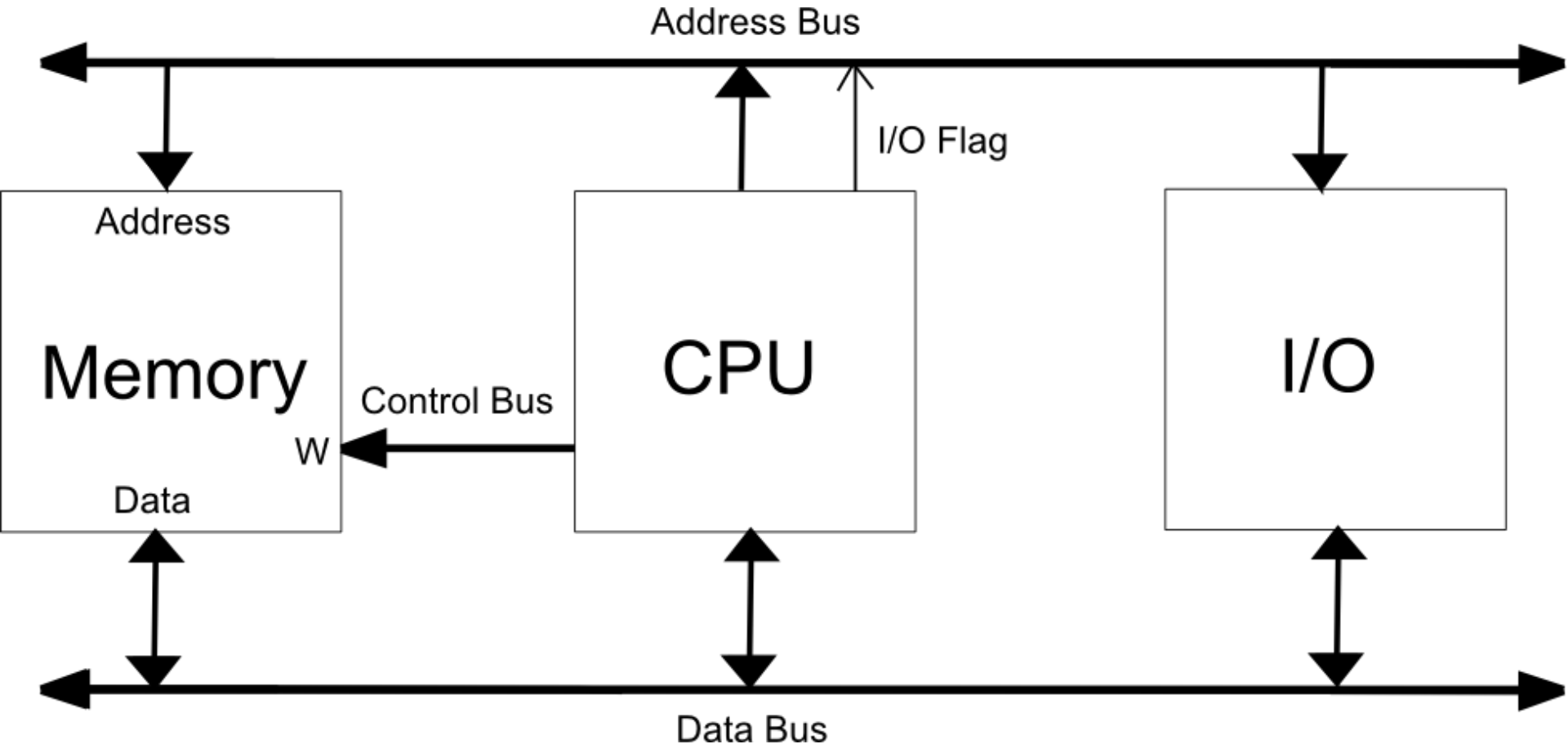
¿Cómo podemos hacer que un programa se comunique con un dispositivo de I/O?



Un programa puede comunicarse con un dispositivo de I/O mediante dos formas: i) memory mapped I/O



Un programa puede comunicarse con un dispositivo de I/O mediante dos formas: i) memory mapped I/O o ii) port I/O



- Una posibilidad: CPU con una única instrucción de input y una única instrucción de output:
  - cada instrucción selecciona uno de los dispositivos
  - un único carácter es transferido entre un registro fijo y el dispositivo
  - el procesador ejecuta una secuencia fija de instrucciones por cada carácter escrito o leido
- Memory-mapped I/O:
  - los registros del dispositivo son parte del espacio de direcciones del computador, y son leidos/escritos usando instrucciones normales
- Problema:
  - la CPU no sabe cuándo el dispositivo de I/O estará listo para enviar/recibir, por lo que debe revisarlos continuamente —***polling***, que produce ***busy waiting***

Interacción entre CPU, memoria y dispositivos puede llevarse a situaciones reales

Usaremos como analogía a un curso haciendo una guía de ejercicios:

- Alumnos (I/O) hacen guía de ejercicios, si alguien termina, el profesor quiere guardar la respuesta.
- Profesor (CPU) esta preparando clases y corrigiendo pruebas.
- Pizarrón o proyector (Memoria), donde se pueden ver los ejercicios y anotar las respuestas.
- Comandos: instrucciones de parte del profesor.
- Estado: alumnos trabajando, con dudas.
- Datos: preguntas y respuestas guía, dudas alumnos, respuestas profesor.
- Supuestos: alumnos no hablan entre ellos, se entregan varias guías durante la clase

Revisaremos tres modelos de interacción ente alumnos y profesor

1. Sin proyector ni pizarrón ni copias de las guías, alumnos tímidos.

   Polling

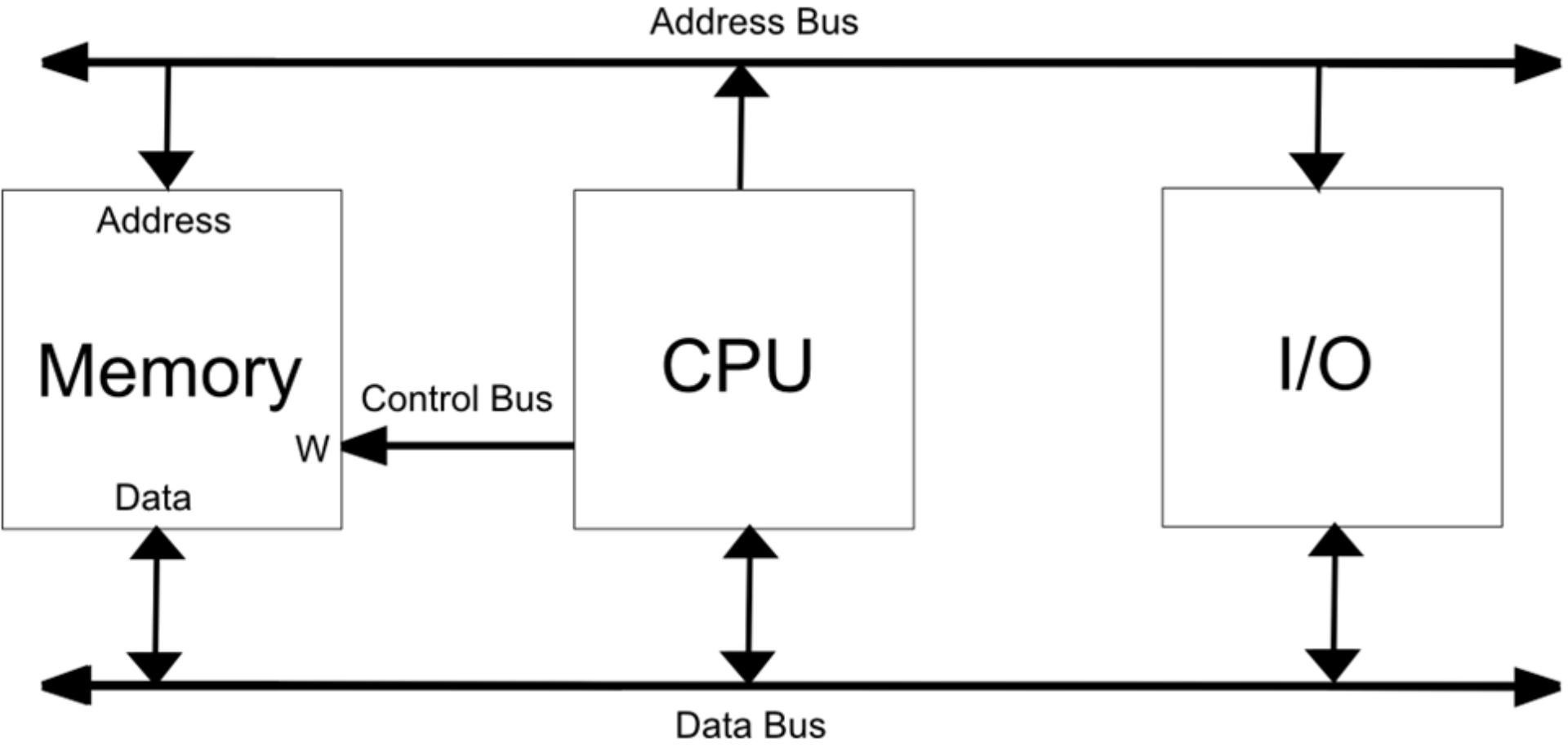
2. Sin proyector ni pizarrón ni copias de las guías, alumnos preguntones y participativos.

   Interrupción

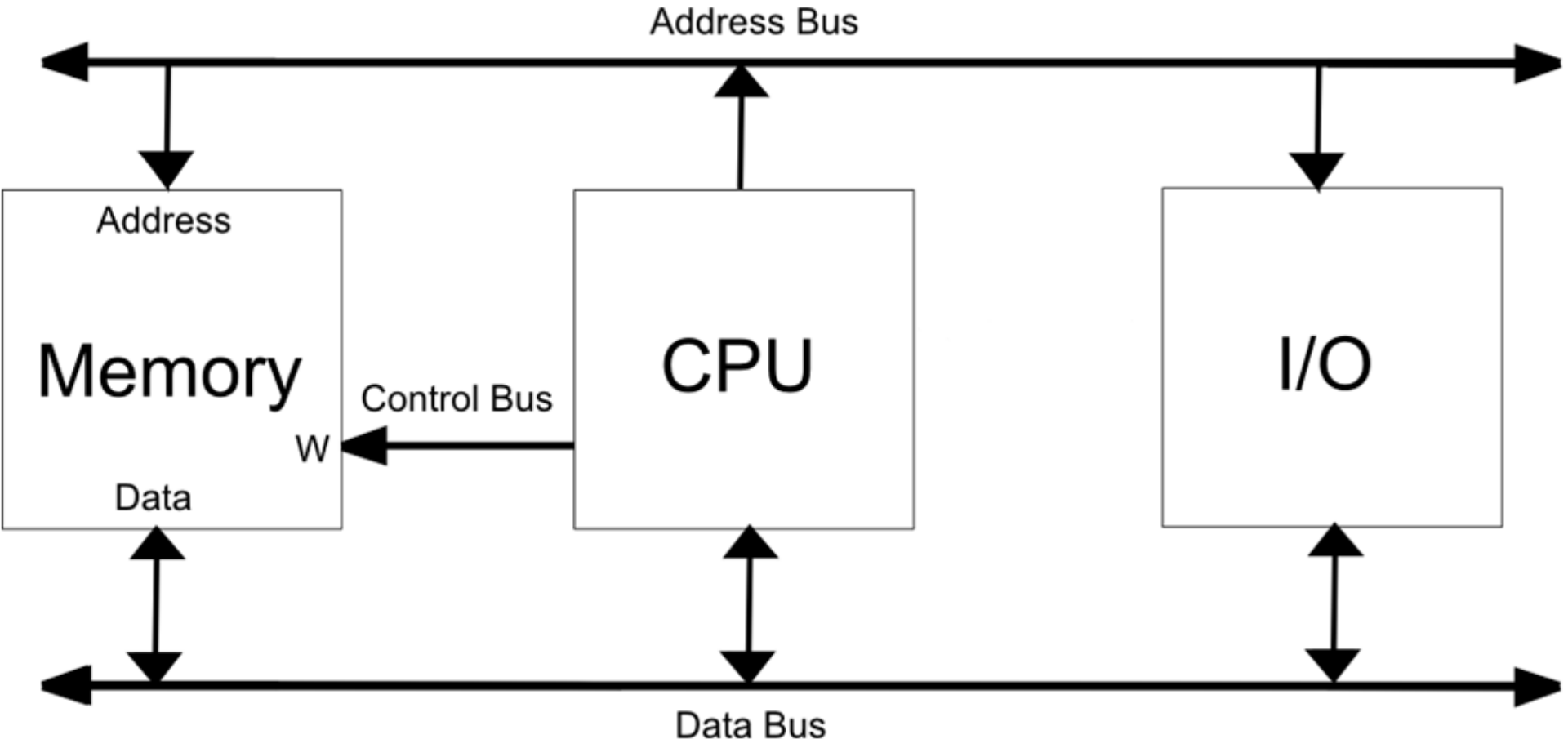
3. Pizarra interactiva, alumnos preguntones.

   Interrupción con acceso directo a memoria

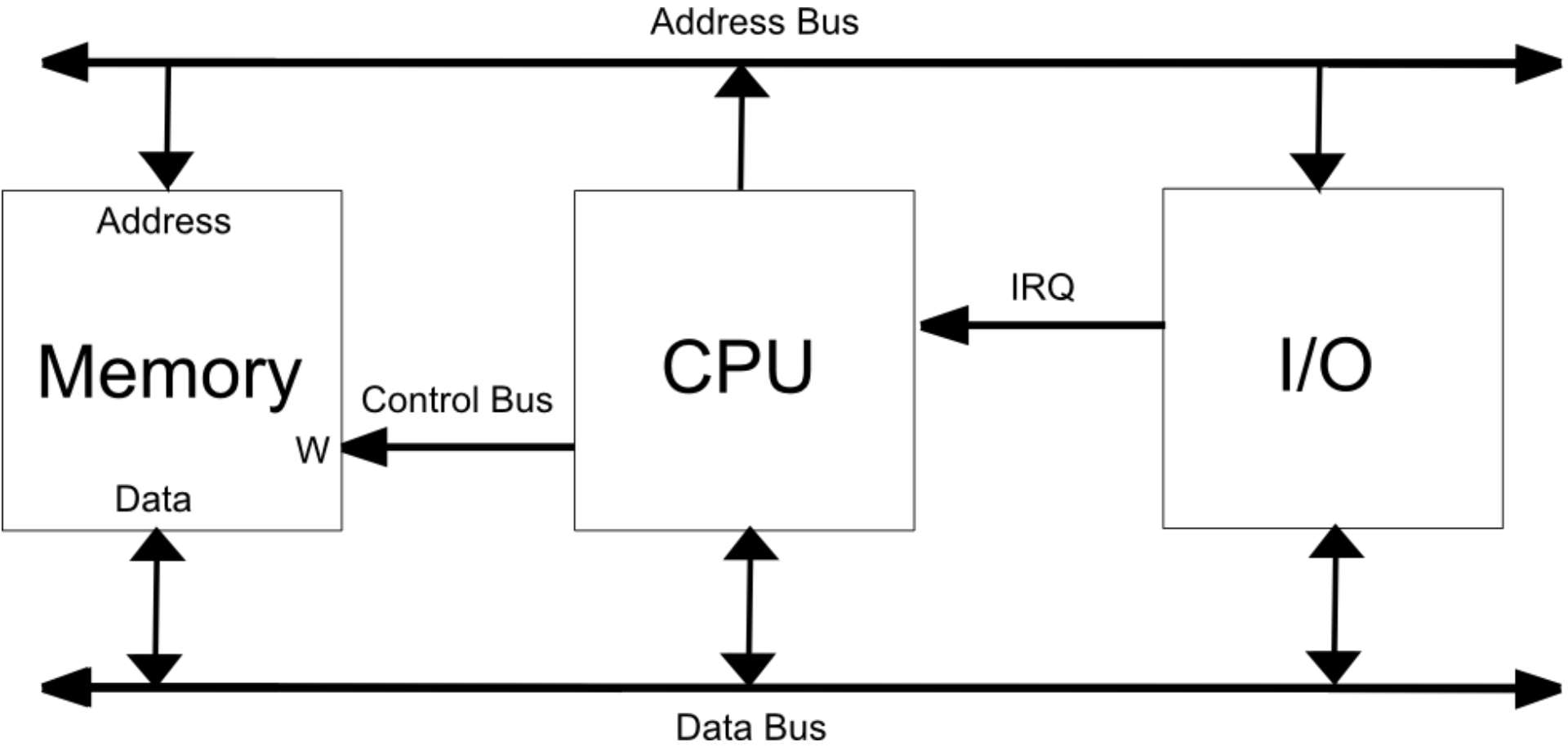
¿Necesitamos nuevo HW para el esquema de polling?
(sin contar Address Decoder e I/O Flag)



¿Cómo podemos mejorar el esquema de polling?

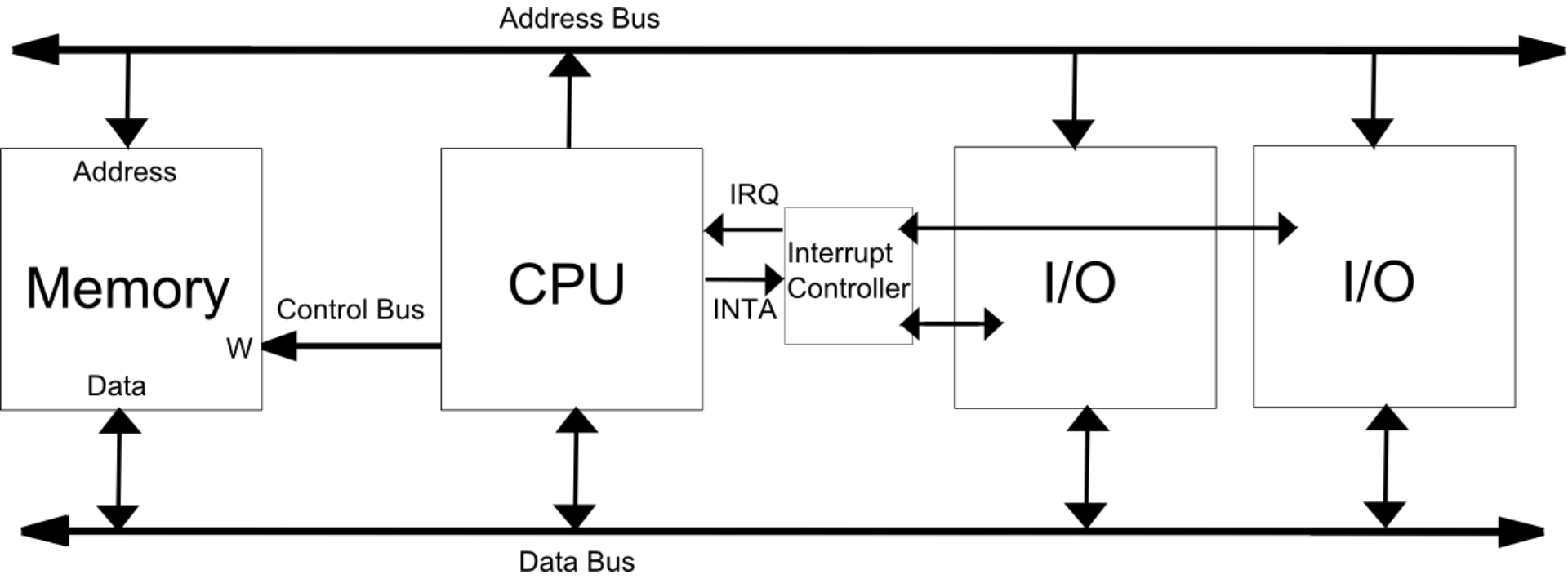


Para evitar la ineficiencia generada por el polling, se agrega la posibilidad de que los dispositivos interrumpan a la CPU



- Una interrupción cambia el flujo de control, pero no debido al programa que se está ejecutando:
  - p.ej., cuando el disco avisa que ha terminado de transferir datos
- La CPU hace partir al dispositivo y le dice que genere una interrupción cuando esté listo:
  - se "setea" un bit *interrupt enable* del dispositivo
- La interrupción detiene el programa y transfiere el control a un manejador de interrupciones:
  - primero ejecuta las acciones apropiadas
  - luego, devuelve el control al programa, que debe ser reanudado en exactamente el mismo estado que tenía cuando ocurrió la interrupción
- Problema:
  - se requiere una interrupción por cada carácter transmitido
- Solución:
  - se puede usar un controlador de interrupciones

Un controlador de interrupciones es una pieza de hardware que se conecta con múltiples dispositivos de I/O por un lado, y con la CPU por otro

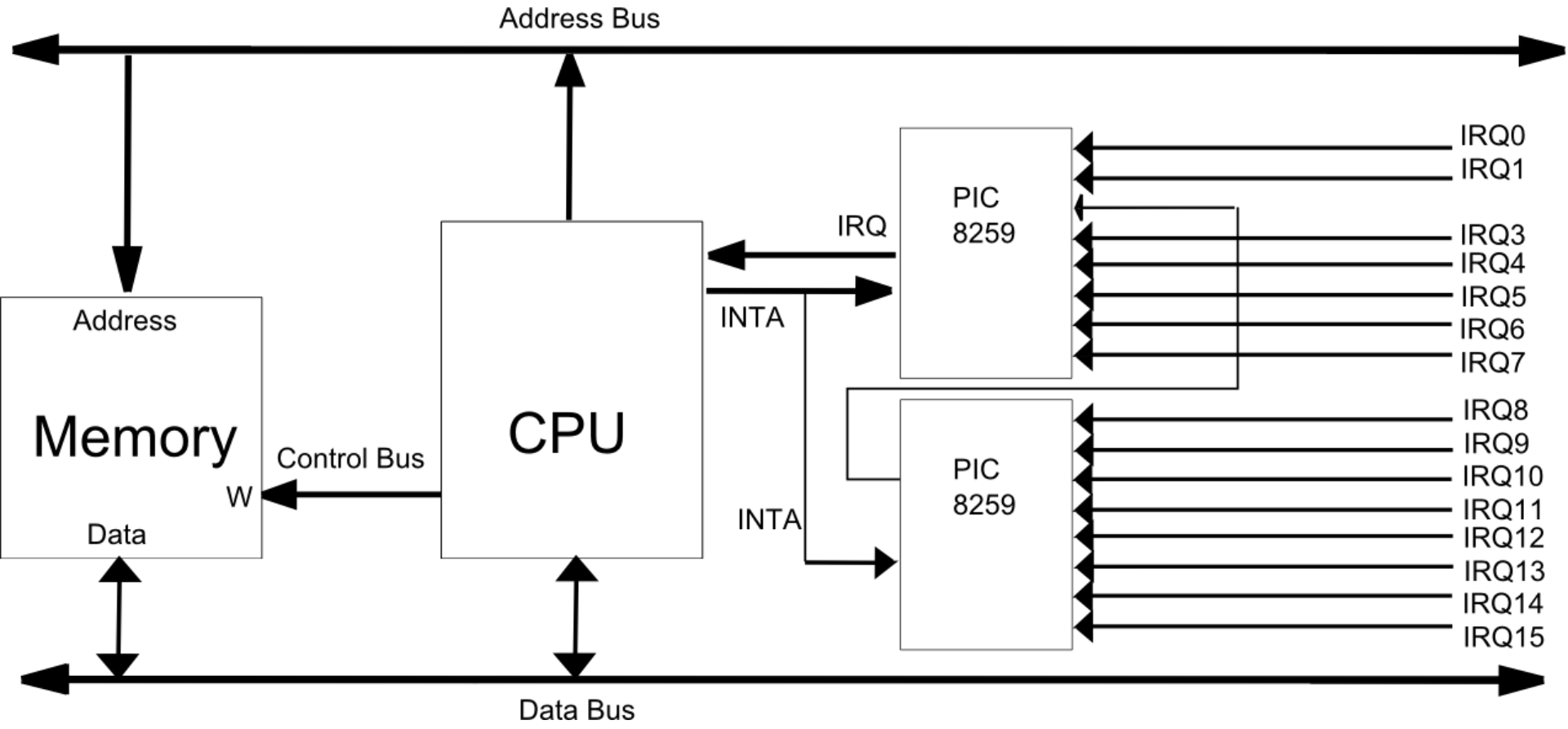


Un controlador de interrupciones es una pieza de hardware que se conecta con múltiples dispositivos de I/O por un lado, y con la CPU por otro

En general, un controlador de interrupciones tendrá al menos los siguientes componentes:

- Registro de comandos y estado
- Registro de interrupciones en espera de atención
- Registro de interrupciones en atención
- Registro de enmascaramiento de interrupciones
- Circuito para manejar prioridades de interrupciones

La arquitectura x86 de 16 bits tiene 2 controladores de interrupciones llamados PIC, los cuales se conectan en cascada



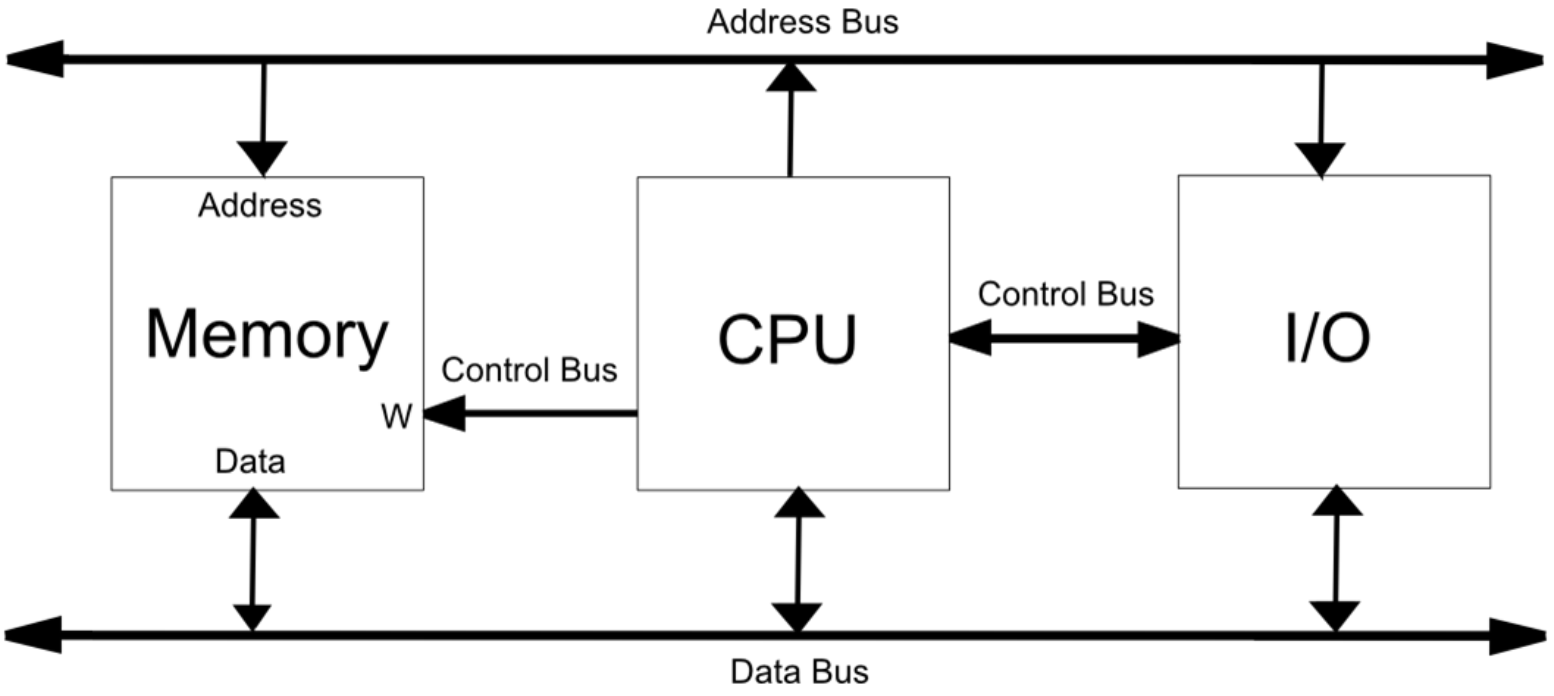
A cada señal de IRQ se le asocia un dispositivo específico y una posición en el vector de interrupciones

IRQ	Dispositivo	Vector de interrupción
IRQ0	Timer del sistema	08
IRQ1	Puerto PS/2: Teclado	09
IRQ2	Conectada al PIC esclavo	0A
IRQ3	Puerto serial	0B
IRQ4	Puerto serial	0C
IRQ5	Puerto paralelo	0D
IRQ6	Floppy disk	0E
IRQ7	Puerto paralelo	0F
IRQ8	Real time clock (RTC)	70
IRQ9-11	No tienen asociación estándar, libre uso.	71-73
IRQ12	Puerto PS/2: Mouse	74
IRQ13	Coprocesador matemático	75
IRQ14	Controlador de disco 1	76
IRQ15	Controlador de disco 2	77

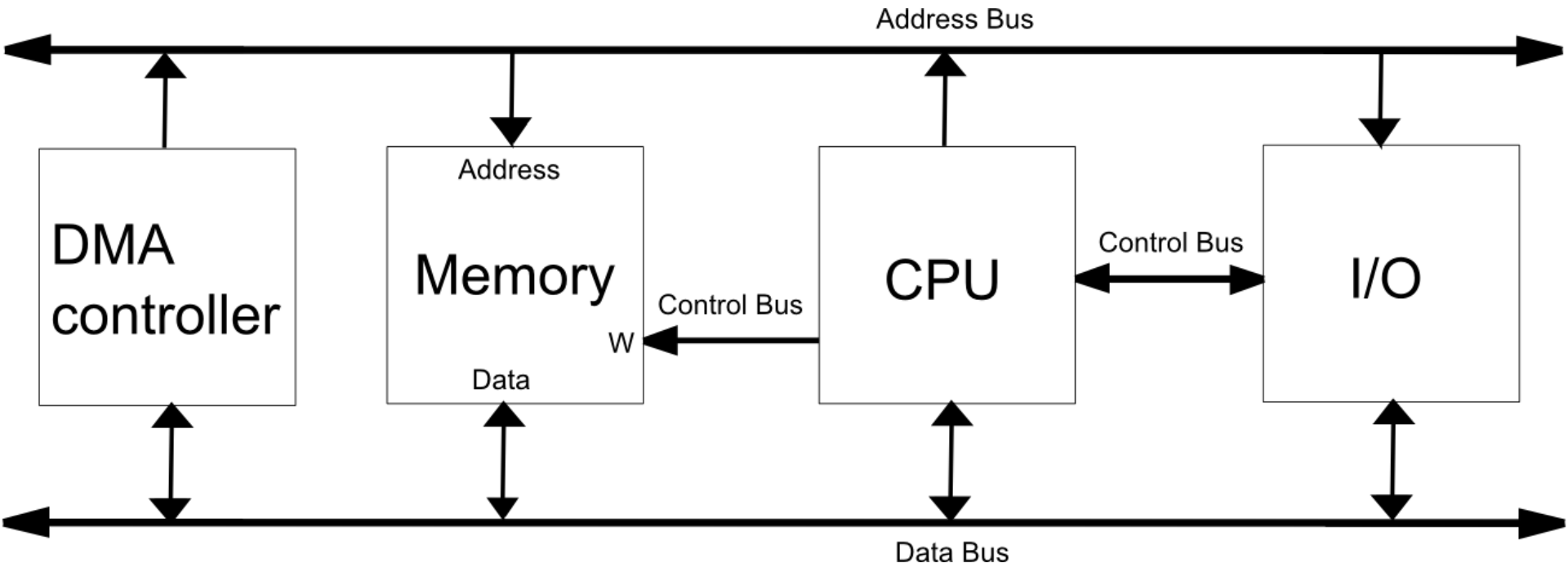
Además de **interrupciones de hardware**, existen las excepciones y las interrupciones de software (traps)

Dirección del vector	Tipo	Función asociada
00-01	Excepción	Excpetion handlers
02	Excepción	Usada para errores críticos del sistema, no enmascara
03-07	Excepción	Excpetion handlers
08	IRQ0	Timer del sistema
09	IRQ1	Puerto PS/2: Teclado
0A	IRQ2	Conectada al PIC esclavo
0B	IRQ3	Puerto serial
0C	IRQ4	Puerto serial
0D	IRQ5	Puerto paralelo
0E	IRQ6	Floppy disk
0F	IRQ7	Puerto paralelo
10	Int. de Software	Funciones de video
11-6F	Int. de Software	Funciones varias
70	IRQ8	Real time clock (RTC)
71 - 73	IRQ9-11	No tienen asociación estándar, libre uso
74	IRQ12	Puerto PS/2: Mouse
75	IRQ13	Coprocesador matemático
76	IRQ14	Controlador de disco 1
77	IRQ15	Controlador de disco 2
78-FF	Int. de Software	Funciones varias

Hasta el momento, todos los datos deben pasar por la CPU para llegar a la memoria o a un dispositivo de I/O (Programmed I/O o PIO)



Para permitir que los I/O tengan **acceso directo a memoria** se utiliza un controlador DMA



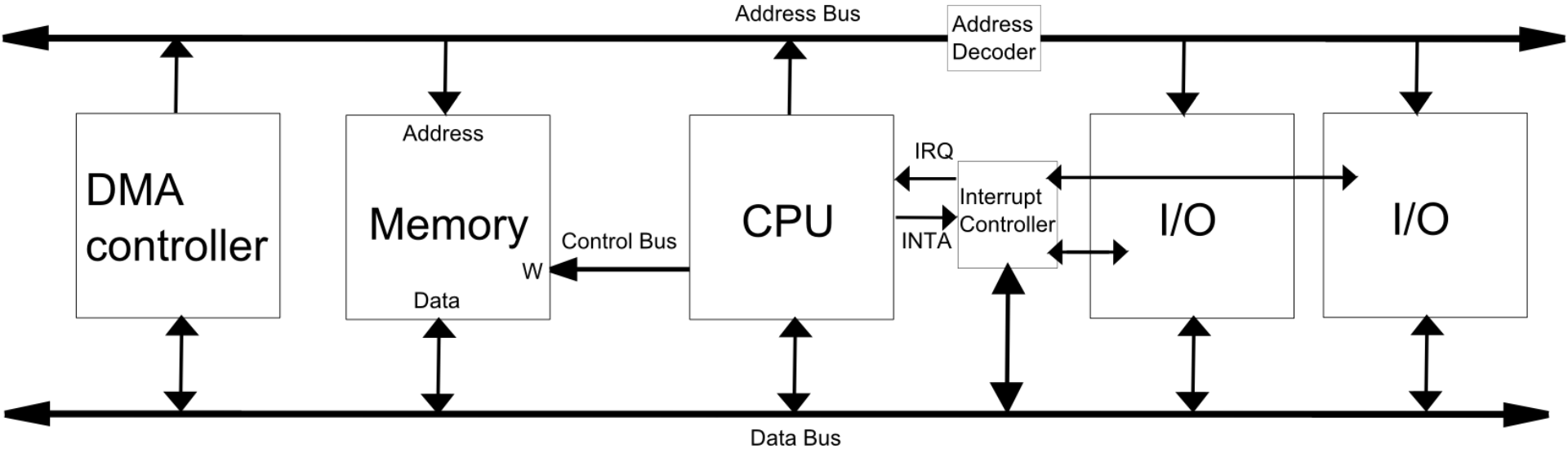
- Agregamos un **controlador DMA**, con acceso directo al bus
- … y con registros que pueden ser cargados por la CPU:
  - la dirección de memoria a leer o escribir
  - número (count) de bytes o palabras a transferir
  - número (identificador) del dispositivo de I/O que se usará
  - dirección (lectura o escritura) de transferencia
- Una vez inicializados los registros, el controlador DMA
  - hace una solicitud al bus, p.ej., para leer el byte en la dirección 100 de la memoria
  - hace una solicitud de I/O al dispositivo correspondiente, p.ej., un terminal, para escribir allí el byte leido
  - incrementa su registro de dirección y decrementa su registro count
  - si count es > 0, repite las operaciones anteriores
  - finalmente, cuando count = 0, **interrumpe** a la CPU

Para permitir que los I/O tengan **acceso directo a memoria** se utiliza un controlador DMA

En general, un controlador de DMA tendrá al menos los siguientes componentes:

- Registro de comandos y estado
- Registros de dirección de origen y destino
- Registro contador de palabras
- Buffer de almacenamiento temporal
- Unidad de control

Arquitectura de un computador con memory mapped I/O, interrupciones y DMA

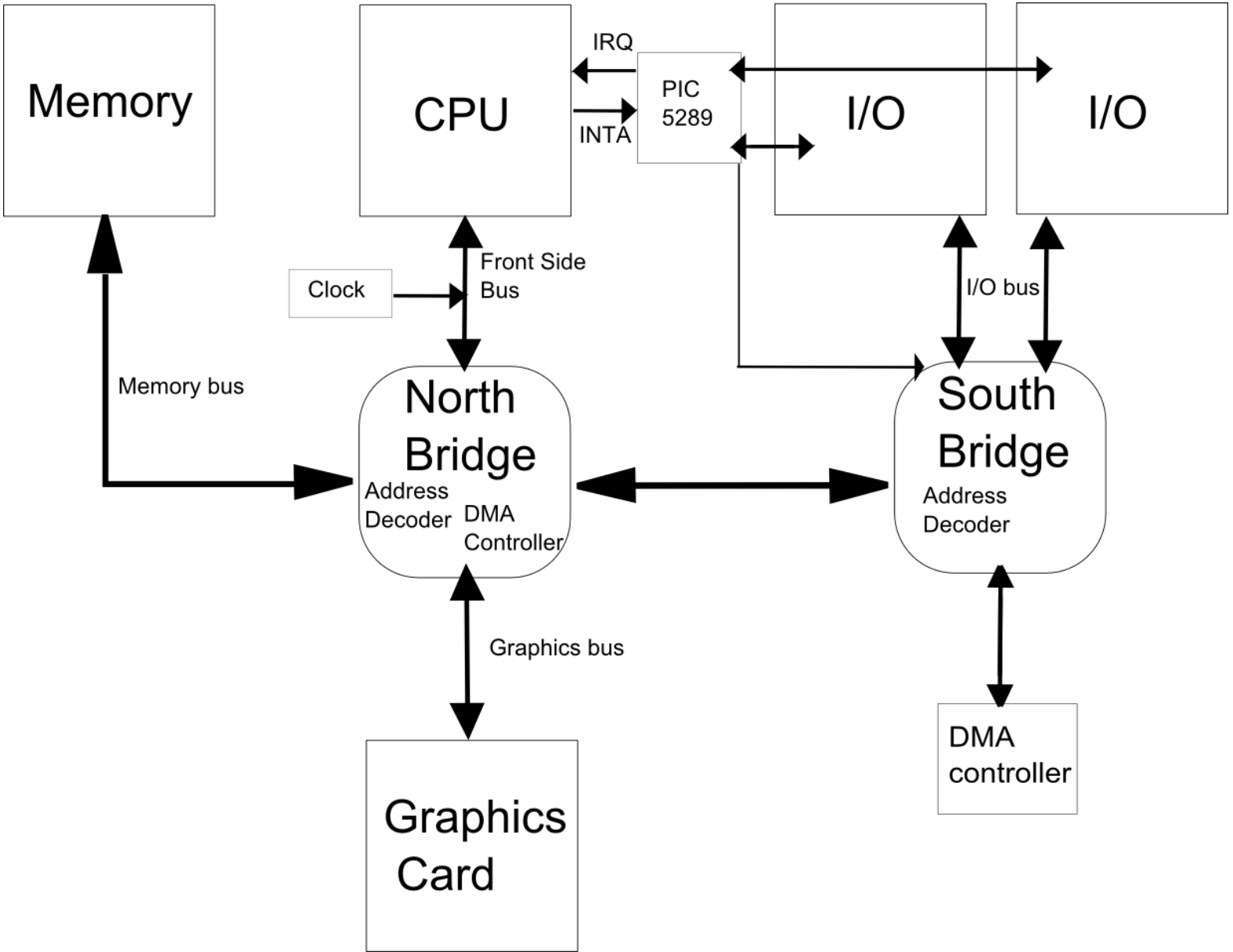


Los dispositivos de I/O pueden organizarse en base a 3 características: comportamiento, comunicante y velocidad de transferencia

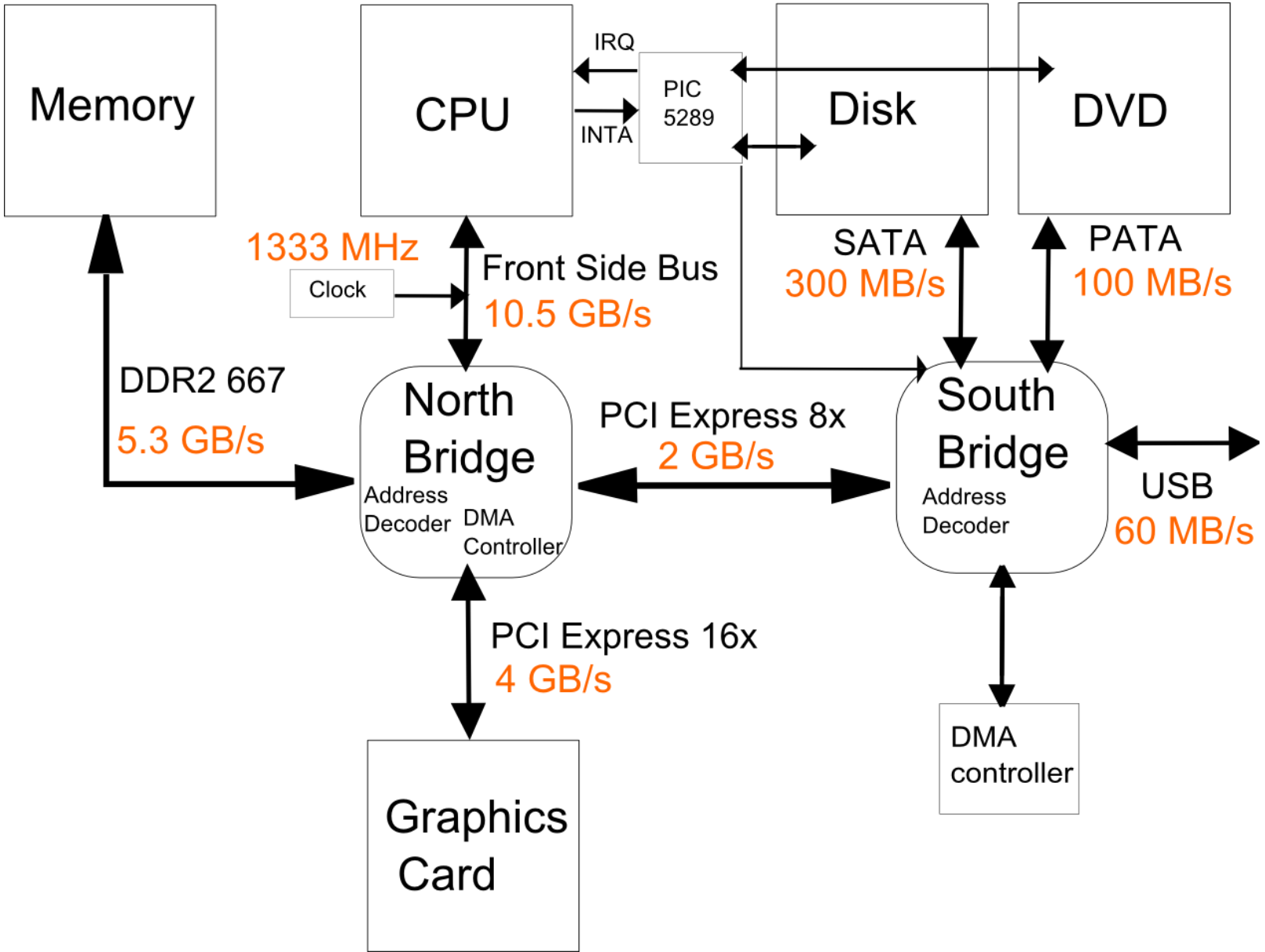
Dispositivo	Comportamiento	Comunicante	Velocidad de transferencia (Mbit/seg)
Teclado	Input	Humano	0.0001
Mouse	Input	Humano	0.0038
Voice in	Input	Humano	0.2640
Scanner	Input	Humano	3.2
Voice out	Output	Humano	0.2640
Impresora laser	Output	Humano	3.2
Display	Output	Humano	800
Red por cable	Input y Output	Máquina	100-1000
Red inalámbrica	Input y Output	Máquina	11-54
Disco óptico	Almacenamiento	Máquina	80-220
Disco magnético	Almacenamiento	Máquina	800-3000

¿Cómo manejamos las distintas velocidades de los participantes en la comunicación?

Arquitectura de un computador **x86** con memory mapped I/O, interrupciones y DMA



El chipset del computador es el conjunto de circuitos ubicados en la placa madre encargado de conectar las distintas partes



¿Y qué pasa entonces cuando prendemos un computador?

- Todos los computadores tienen un programa de inicio cargado en una ROM.
- Inicialmente BIOS, actualmente UEFI (*unified extensible firmware interface*).
- Revisa y activa hardware del computador.
- Carga sección de inicio del SO en memoria.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores

Conexión de CPU y Memoria con I/O

Profesor: Hans-Albert Löbel